Heriot-Watt University

Masters Thesis

Building Trust in Explanations of Machine Learning Models

Author:
Michael Pidgeon (H00360036)

Supervisor:
Prof. Ekaterina Komendantskaya

August 2021

Declaration of Authorship

I, Michael Pidgeon (H00360036), declare that this thesis titled, 'Building Trust in Explanations of Machine Learning Models' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Michael Pidgeon

Date: 23 August 2021

Abstract

One challenge with machine learning is the models used are often 'black boxes', which cannot be meaningfully interpreted by humans. A number of 'post-hoc' explainable AI techniques can help provide a better understanding of why certain predictions were made, without requiring access to model internals.

This project investigated metrics that can be applied to these post-hoc explanations, in order to build confidence in their robustness. In the context of explanations, robustness means ensuring small input perturbations do not cause major changes in the explanation, as well as assessing its degree of faithfulness to the underlying model.

These two robustness characteristics were assessed by a novel implementation of explanation 'sensitivity' and 'infidelity' metrics. This implementation was published as an open-source Python package, with the results successfully tested and validated.

The metrics were used to provide useful information on explanations of a series of 'real-world' models, whose underlying robustness is uncertain. The metrics showed a clear ability to detect overfitted models, as well as some evidence of detecting adversarial models (where the explanations are deliberately manipulated).

My overall aim in this project was to add to the body of work examining methods of building trust in explanations of machine learning models - and, by implication, help understand when such explanations should *not* be trusted. I feel the project has met this aim, and the open-source software published as a result should prove useful. Further work could provide a better understanding of the subtleties in applying the metrics in practical situations.

Acknowledgements

I would like to thank Prof. Ekaterina Komendantskaya for being an excellent supervisor. She has taken a real interest in my work, been patient with my less intelligent questions, and answered my occasional more intelligent questions with expert insight. I particularly appreciate the effort she has made during this challenging and busy year.

I would also like to thank others in the Lab for AI Verification - in particular those who provided constructive feedback on my talk covering this subject, and Dr Matthew Daggitt who has been a huge help throughout.

Finally, I would like to thank the second reader of this report, Prof. Lynne Baillie, for her helpful feedback on document structure at the Research Report stage.

Contents

List of Figures

List of Tables

Abbreviations

AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
DNN	**D**eep **N**eural **N**etwork
EXRT	**EX**planation **R**obustness **T**oolbox
FGSM	**F**ast **G**radient **S**ign **M**ethod
LIME	**L**ocal **I**nterpretable **M**odel-agnostic **E**xplanations
ML	**M**achine **L**earning
MLP	**M**ulti **L**ayer **P**erceptron
SHAP	**SH**apley **A**dditive ex**P**lanations
XAI	e**X**plainable **AI**

Chapter 1

Introduction

1.1 Background

Recent advances in machine learning (ML) and artificial intelligence (AI) have had a major impact on society and business. ML techniques are used in an ever-increasing number of domains: from self-driving cars to preventative policing. Such systems have been shown to outperform humans in certain domains, including game playing (Silver et al., 2016) and image classification (Krizhevsky et al., 2012). One accountancy firm estimates these techniques will drive GDP gains of over $15 trillion by 2030 (PwC, 2018).

Many ML techniques, such as deep and convolutional neural networks (DNNs/CNNs) involve decisions being made by 'black-box' models that are not interpretable by humans. This can be either because their internals are obscured, or are simply too complex to understand.

A variety of 'explainable AI' (XAI) techniques offer to help with this problem. The number of such techniques has proliferated in recent years, as has interest in the concept in general (see Figure 1.1). One popular class is that of the 'post-hoc' explanation (see Figure 1.2). This involves treating the model as a black box, and attempting to understand its behaviour by probing the relationship between its inputs and outputs.

However, it is generally not possible to fully reverse engineer a model in this way. Indeed, many such techniques involve building a 'surrogate model' which is inherently lacking

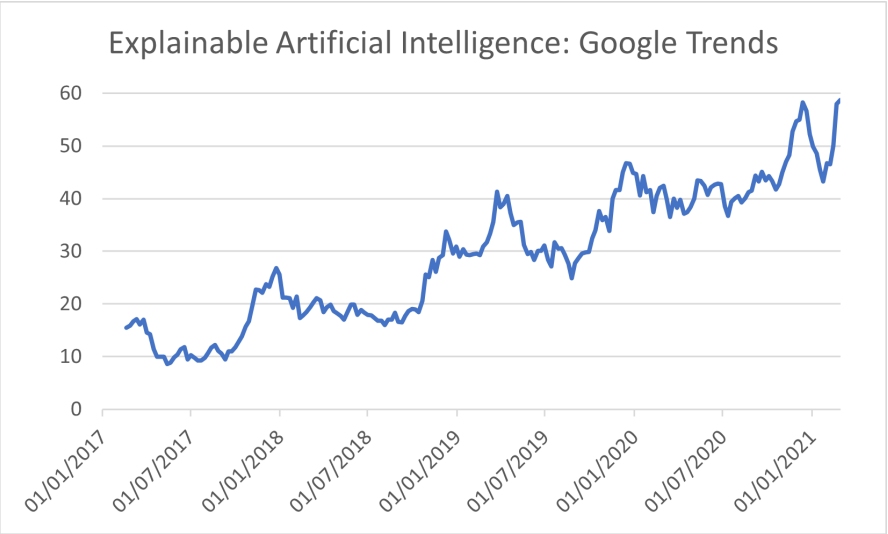

FIGURE 1.1: Google Trends data showing increase in web searches for 'Explainable Artificial Intelligence' since 2017. Weekly data, shown as 8-week moving average; 0 = insufficient data; 100 = weekly maximum during the period.

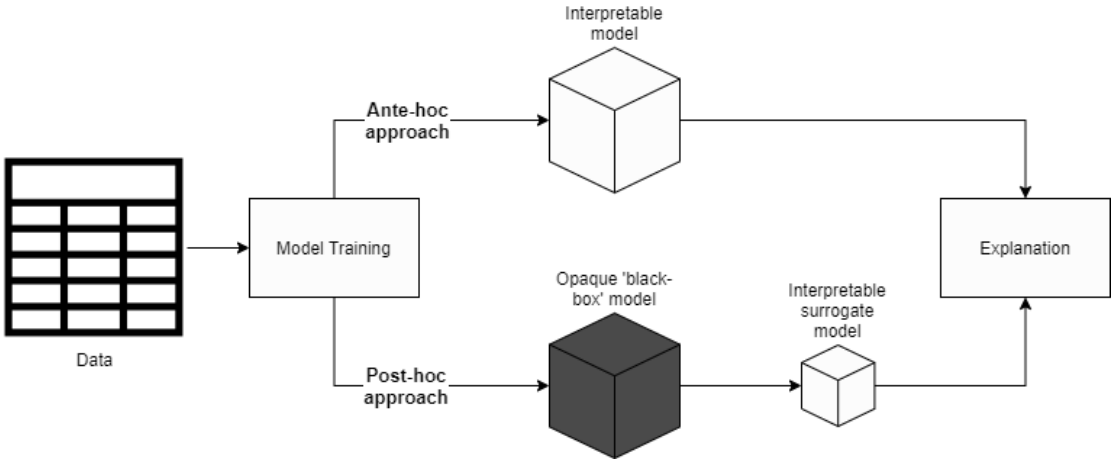

FIGURE 1.2: Summary of the difference between ante-hoc and post-hoc/surrogate model approaches to generating explanations. Popular post-hoc techniques such as LIME (Ribeiro et al., 2016) can be used to generate an interpretable model from the original, black-box model.

full alignment with the original model (Rudin, 2018), as full alignment would imply equivalence between the surrogate and original models.

In addition, recent work has shown explanations can be fooled by an adversary, in a manner similar to the better-known approach of an adversarial attack on the model itself (Goodfellow et al., 2015, Szegedy et al., 2014). Such attacks allow an explanation that is deliberately misleading to be generated, without affecting the underlying model predictions (Slack et al., 2019).

1.2 Motivation

This project aimed to add to the existing body of work studying how post-hoc explanations can be assessed for robustness.

Considering explanation robustness is an inherently challenging subject, as one needs to make a distinction between this and model robustness. The definitions for both can vary in the literature - (Casadio et al., 2021) divides model robustness into four formally defined categories, although the ML literature is sometimes not clear which type is being referred to.

In this paper, I divide explanation robustness into two slightly different variants - '**sensitivity**' (level of explanation change caused by a small input change), and '**infidelity**' (level of faithfulness to the underlying model). These metrics are formally defined in (Yeh et al., 2019). This work provided a helpful starting point for this project, however it focusses on the theoretical links between the metrics and explanations, and does not consider what useful information they can provide about real-world models.

While the authors of this paper published their code, it is tightly coupled to their experiments and cannot be deployed to arbitrary models and explanations. There is currently no such tool available for assessing the robustness of explanations. Such a tool would be a valuable contribution, given the increasing level of interest in XAI.

1.3 Aims and Objectives

My overall aim in this project was to assess and improve existing methods of measuring the robustness of ML explanations, and publish original open-source software allowing these measurements to be more easily applied by others. My hope was such software may prove useful, for example, when meeting regulatory requirements (e.g. GRPR's 'right to an explanation' (Goodman and Flaxman, 2017)), when dealing with sensitive personal data, or when concerns around bias are raised.

The specific objectives I proposed at the outset, and a short summary of whether they were met, is given in Table 1.1.

All objectives were met, albeit with one having slightly reduced scope due to time constraints. One (Objective 5) was expanded to include an evaluation of an underlying mathematical property (completeness) of explanation techniques, see Section 5.3.

This thesis has resulted in working software being published, which I hope to build on in future as an open-source project - see `https://pypi.org/project/exrt/`.

Section 3.3 gives a more detailed view of these objectives, split into prioritised requirements.

Further detail of experimental objectives is provided by my hypotheses (refer to Section 3.2).

TABLE 1.1: Summary of objectives

Objective	Met?
Objective 1. Identify two suitable datasets containing nominal data, suitable for classification problems. Datasets should present a contrast, e.g. balanced vs unbalanced.	✓ 'Bank Marketing' and 'Spambase' datasets selected, refer to Section 4.3.
Objective 2. For each dataset, use TensorFlow (Abadi et al., 2016) to train: • A 'baseline', high-performing model to serve as the control. • An 'adversarial' model which generates deliberately false explanations, using the techniques described in (Slack et al., 2019). • A 'sub-optimal' model which gives poor classification/regression performance, without using any adversarial techniques (e.g. due to overfitting).	✓(reduced scope) Baseline models trained for both datasets. Adversarial model trained for Spambase dataset only. Overfitted model trained for Bank Marketing only. Refer to Section 4.3.
Objective 3. Use at least one post-hoc explanation technique to generate local explanations for a sample of test-set inputs for each of our models.	✓ SHAP (Lundberg and Lee, 2017) used to generate local explanations for test-set inputs for all models. Refer to Section 4.3.
Objective 4. Implement explanation robustness metrics in a way that is model and explanation-agnostic, and can be applied to categorical as well as numerical data.	✓ Novel implementation of metrics proposed by (Yeh et al., 2019) written, validated and published as an open-source package. Refer to Section 4.1.
Objective 5. Carry out empirical and mathematical validation of the metrics.	✓ Metrics successfully validated using a variety of techniques, refer to Section 5.2
Objective 6. Assess the explanations generated in Objective 3 against these metrics.	✓ Results compared using Hypotheses 3 and 4. Refer to Section 5.4 and Section 5.5.
Objective 7. Publish an open-source tool that can be used to quickly and simply assess post-hoc explanation robustness.	✓ Package published to PyPI (PYpI, 2021) Python package repository, refer to Section 4.2.

Chapter 2

Literature Review

2.1 Background

2.1.1 Machine Learning

Machine learning (ML) is a branch of artificial intelligence (AI). In the most general sense, ML is the use of statistics to train a model which makes predictions based upon patterns in data.

Two common approaches to training such a model are:

- **Supervised**: learning from labelled training data, and assuming the model generalises to unseen data. Most commonly used for classification problems - can also be used for regression.
- **Unsupervised**: learning patterns in unlabelled data. Often used in the context of clustering algorithms such as k-means.

Reinforcement learning falls outside the above definition, and will not be covered in this project.

Much recent work has focused on artificial neural networks (ANNs), and specifically deep neural networks (DNNs) (LeCun et al., 1989). These networks build on the Perceptron model (Rosenblatt, 1958). They are loosely modelled on synapses in the human brain,

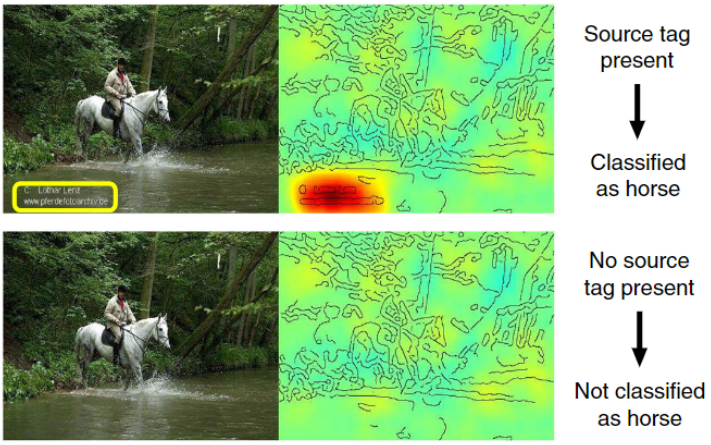

FIGURE 2.1: Example of a neural network learning surface statistical regularities from (Samek, 2019). The model has trained itself to recognise the text in the bottom-left of the image, and associate it with horses. It has not learnt the 'true abstractions' of a horse (body shape, mane etc) in the manner of a human.

and can use non-linear activation functions to model non-linear relationships. They have shown excellent performance for a number of tasks, such as image classification using convolutional neural networks (CNNs) (Lecun et al., 2015).

However - DNNs and CNNs have two important drawbacks:

- Their strong performance in certain domains comes at the cost of low interpretability. They are viewed as 'black-boxes' - not necessarily because their internals are inaccessible, but because the workings of a non-trivial ANN are too complex for a human being to reason with.
- They exhibit a tendency to learn surface statistical regularities, rather than true abstractions (Jo and Bengio, 2017). This manifests itself in models learning to pick up on irrelevant features (see Figure 2.1), and is the underlying reason for the vulnerability of DNNs to adversarial attacks (see Section 2.1.2).

Many ML approaches other than DNNs/CNNs exist. They benefit from using simpler - and therefore more interpretable - algorithms. Examples of these approaches include Bayes nets, decision trees and logistic/linear regression. (Rudin, 2018) makes a compelling argument that such approaches can offer similar real-world performance to DNNs in many domains, due to their interpretability offering insights into how they work, which in turn allows further refinement of the model.

FIGURE 2.2: The first demonstration of adversarial attacks, from (Szegedy et al., 2014). Left hand column showing the original image; centre column showing perturbations; right hand column showing the adversarial image, in each case classified as *ostrich, Strutio camelus*. The change from right → left is not easily perceptible to a human.

2.1.2 Adversarial Attacks on ML Models

(Szegedy et al., 2014) found DNNs have a tendency to learn solutions with counter-intuitive properties, identifying the same phenomenon as (Jo and Bengio, 2017). The former paper was the first to exploit this by applying small perturbations to the input, and observing this had a significant effect on the output - an 'adversarial attack' (see Figure 2.2. They also recognised the same attack had highly similar effects on different networks, trained using different subsets of the data. This demonstrated the same surface statistical regularities were being relied upon across the different networks.

(Szegedy et al., 2014) was not clear on *why* adversarial attacks were possible. (Goodfellow et al., 2015) helped answer this question, by demonstrating the largely linear nature of DNNs made the attacks possible. This paper also gave a computationally efficient way of generating adversarial examples: the fast gradient sign method (FGSM).

This work made generating adversarial examples straightforward, resulting in a large volume of further work. Many other methods of generating such examples have been published, with one interesting approach (Zhao et al., 2017), focusing on examples that are less contrived (i.e. modifying a single pixel) and more natural - see Figure 2.3.

(a) Input instance (b) FGSM Δx (c) FGSM adversary (d) Our Δx (e) Our adversary "2"

FIGURE 2.3: Example of a more natural appearing adversarial example, from (Zhao et al., 2017). The perturbations in image (e) are more difficult for a human to spot than for (c).

Other research in this field has focused on two intriguing problems:

1. **Real-world relevance of adversarial attacks**. Sign recognition for self-driving cars has often been used as a case study (Sitawarin et al., 2018), with the overall conclusion that adversarial attacks are possible in a real-world setting.

2. **Defence techniques**. Many techniques have been proposed to improve resilience to adversarial attacks, the most common being adversarial training (Kurakin et al., 2016), where the training set is supplemented with adversarial examples. Some work has found there is an inherent trade-off between adversarial robustness and accuracy (Tsipras et al., 2018), with one study even finding adversarial training can compromise safety in robotics applications (Lechner et al., 2021). This trade-off is disputed by other research - in particular, recent techniques involving optimising saliency of Jacobian matrices seem to have promise in improving both robustness and accuracy (Chan et al., 2019).

For this project, my interest in adversarial attacks is twofold:

1. **Their theoretical links to explanations**. (Ignatiev et al., 2019) demonstrates a theoretical basis for their relationship. It achieves this by simplifying the idea of an explanation to an 'absolute explanation' - an input space that will always give the same classification result. First order logic is then used to demonstrate the theoretical connection, although experimental evidence is limited to a proof of concept only.

2. **The parallels between adversarial attacks on model outputs, and attacks on the explanations of these outputs**. Some 'explanation attacks' use similar techniques, involving calculation of optimal input perturbations (Dombrowski et al., 2019), see Section 2.2 for further details.

2.1.3 Neural Network Verification

Formal verification of non-ML software is a mature research area, with many well-developed and scalable 'theorem provers' available to prove a program's correctness (D'Silva et al., 2008). Although such techniques are not routinely used in industry, there are many examples of them being successfully applied to real-life, complex and fast-moving software projects (Calcagno et al., 2015).

Automatic theorem provers are the most commonly used for software verification. This type of theorem proving does not require human input once properties to prove, and the model itself, have been defined. (The other category, interactive theorem provers, is focused on human-machine collaboration).

Most automatic theorem provers are *sound*, meaning they will only ever report that a property holds if this assertion is correct. Some techniques are also *complete*, meaning that whenever the property holds, the algorithm will correctly state that it holds (Liu et al., 2019). Satisfiability modulo theories (SMT) are often used for the implementation of such provers. SMT solvers convert problems to classical first-order logic, and then assesss for satisfiability (for example, we can trivially see $x > 5$ would be satisfied by any value of x greater than 5).

Automatic theorem provers have been applied to ANN and DNN models. In a real-world setting, this is challenging: (Komendantskaya et al., 2020) gives three reasons for this:

1. **Inability to decompose**. (D'Silva et al., 2008) details traditional verification techniques being used to assert against specific parts of the code that matter, for example checking no 'divide by zero' errors are possible. However this is challenging for DNNs - all parts of the network may matter. XAI techniques focused on network internals could theoretically help identify the salient parts of the network - this knowledge could then be used by verification tools (this is a possible approach for my Research Question **??**.

2. **Scalability**. Even a moderate sized DNN may have hundreds of thousands of properties and inputs. Large networks may have numbers of properties that are orders of magnitude higher still. This scale will overwhelm most SMT solvers, although some specialised solvers are available that show promising scalability characteristics.

3. **Non-linear arithmetic**. Non-linear activation functions mean a DNN is itself a non-linear function. There are two options for handling this: either a specialist SMT solver can be used that can handle non-linear arithmetic. Or, activation functions can be linearised - converted to piece-wise linear functions, at the cost of a small loss of accuracy (Kokke et al., 2020).

In order to use any verification technique, one or more properties to verify must be formulated. One broad category of properties can be formulated as a statement that if the input belongs to some set $\mathcal{X}$, the output will belong to some set $\mathcal{Y}$ (Liu et al., 2019). Two primary methods exist to formulate these properties:

1. **Model robustness**. Sometimes referred to as 'pointwise robustness'. Perturbed inputs close to the original inputs should give the same output. An epsilon ball $\mathbb{B}(\hat{x})$ is defined around a sample input $\hat{x}$ of that class: $\mathbb{B}(\hat{x}, \varepsilon) = \{x \in \mathbb{R}^n : ||\hat{x} - x|| \leq \varepsilon\}$ (Komendantskaya et al., 2020) (see Figure 2.4).

2. **Known constraints**. It may be possible to establish such constraints by inspection, or by a priori knowledge for certain boundaries. We may know certain classification predictions should never be made - they are out of bounds under certain input conditions. Or, a safety-critical system may have clearly defined operating limits (for example, (Julian et al., 2019) describes successfully verifying an aircraft collision detection avoidance ANN, and formally proving it could never allow a collision to take place).

Marabou (Katz et al., 2019) is a specialised SMT solver (refer to Figure 2.5 for an architecture summary) which offers several practical advantages for DNN verification:

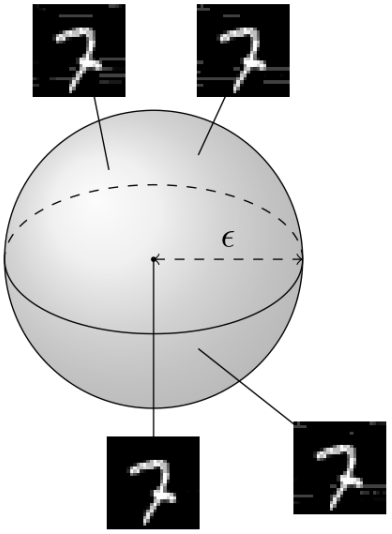

FIGURE 2.4: Epsilon ball concept, diagram from (Casadio et al., 2021). Inputs are perturbed within some ε distance. This approach is used to formally verify points near the training example will belong to the same class as the example.

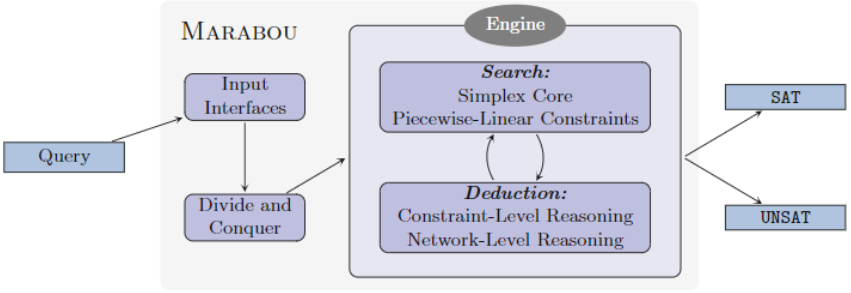

FIGURE 2.5: Summary of Marabou's architecture, from (Katz et al., 2019)

- **Scalability**. Using a combination of high-level reasoning about the properties of the network, heuristics for treating constraints lazily if possible, and parallelisation of execution, Marabou performs well against known benchmarks for verifying DNNs.

- **Provision of counter examples**. If a verification property is found to be unsatisfiable, a counter-example is provided. Intuitively, this feels likely to be highly valuable in this project, a part of which is attempting to use formal verification in the context of XAI techniques.

- **Multiple interfaces**. A TensorFlow model can be imported in protocol buffer format. Python bindings are available to simplify working with the native C++ code.

In this project, my interest in formal verification is its links to XAI. (Ignatiev, 2020) provides a useful summary of work into this area, dividing approaches to generating explanations (or validating other explanations) using formal verification into two categories. *Knowledge compilation* aims at finding simpler canonical representations of models, while preserving their logical integrity. As might be expected, such a process is computationally expensive. *Abductive reasoning* calculates an explanation on an instance-by-instance basis by converting the model to a logic-based representation. These two techniques are analogous to the 'global' and 'local' split of XAI techniques referred to in Section 2.1.4.

2.1.4 Explainable and Interpretable AI

The terms 'interpretable AI' and 'explainable AI' are widely used in the literature. Unfortunately there is no consensus on the definition of either term.

I believe the widely cited (Rudin, 2018) offers the offers the most coherent and useful definitions. Essentially - interpretability is an inherent property of the model, whereas explainability refers to using a separate technique to explain the model in some way.

As explainability techniques often use an interpretable model as a surrogate, this is a valuable distinction to draw. In the remainder of this report, I will use the term explainable/explainability unless referring specifically to an inherently interpretable model.

Explainable AI is not a new idea. Its history goes back to expert systems in the 1970s. It was considered critical for such systems to be able to explain their decisions. For example, (Carbonell, 1970) considered how explanations from an AI tutoring system could be used for pedagogical purposes.

With the proliferation of 'black-box' DNNs in recent years, however, the urgency of finding satisfactory XAI techniques has increased. This has resulted in a large and growing range of techniques. One challenge for an aspiring XAI researcher is to develop a mental model to classify these different techniques. Taxonomies presented in review papers are helpful but often overly complex, and lacking clearly segmented categories. I have found the most useful taxonomy to be that presented in (Yang et al., 2019). A modified version is presented in Figure 2.6. Categories presented in this diagram are as follows:

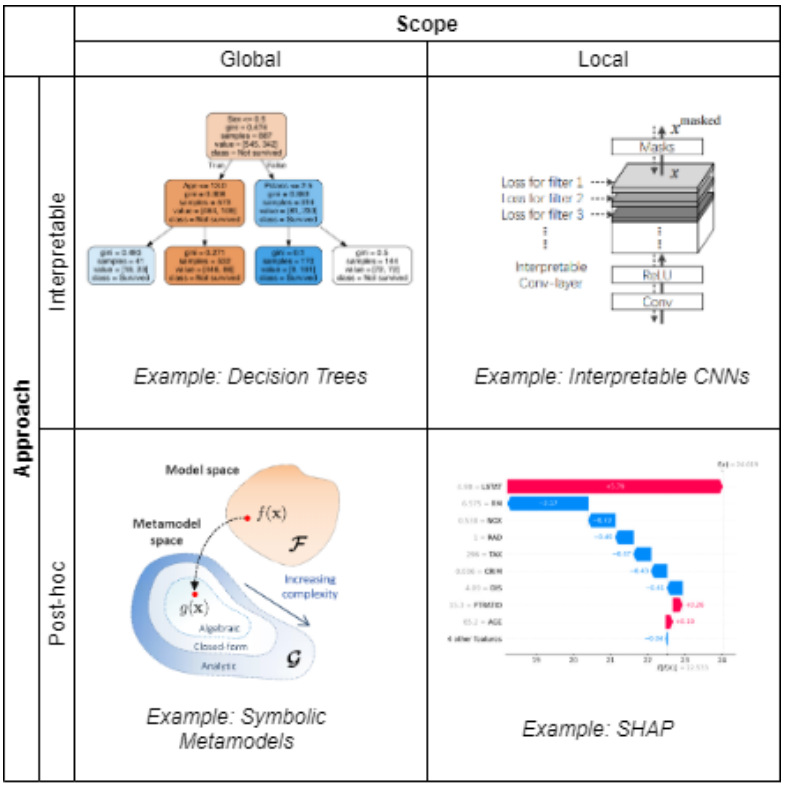

FIGURE 2.6: Summary of XAI taxonomy, adapted from that in (Yang et al., 2019). Image credits - Decison trees: https://towardsdatascience.com/. Interpretable CNNs: (Zhang et al., 2017). Symbolic Metamodels: (Alaa and Schaar, 2019). SHAP: https://github.com/slundberg/shap

- **Interpretable - global**: models which are simple enough for a knowledgeable user to understand without any separate techniques, such as decision trees or linear regression.

- **Interpretable - local**: models which allow an explanation of a single instance to be generated without use of a separate technique, even if the model itself is not interpretable. An example is the 'Interpretable CNN' (Zhang et al., 2017), which represents the abstractions it is using to classify images in a form that can be fed back to the user when new images are encountered.

- **Post hoc - global**: some process is used to simplify a complex model into a simpler, interpretable model. One interesting recent example is (Alaa and Schaar, 2019), which uses Meijer G-functions and gradient descent to train 'models of models' with varying degrees of complexity: see Figure 2.7.

- **Post hoc - local**: an explanation for a single instance is generated, treating the model as a 'black-box' and using some method of systemically comparing inputs to

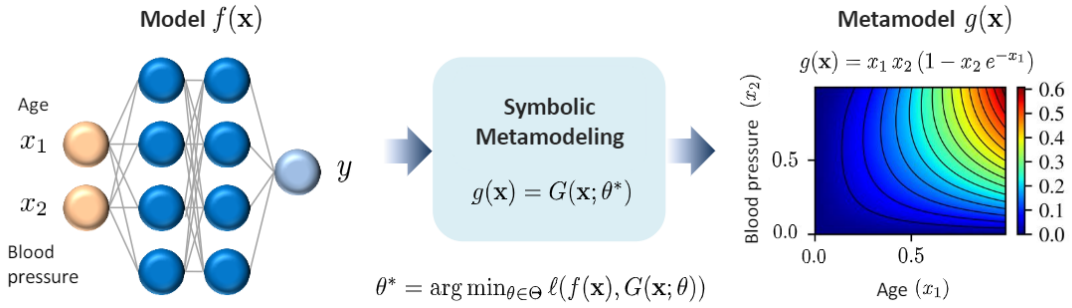

FIGURE 2.7: A summary of the symbolic metamodelling process described in (Alaa and Schaar, 2019). A simple ANN is converted, to an inherently interpretable metamodel.

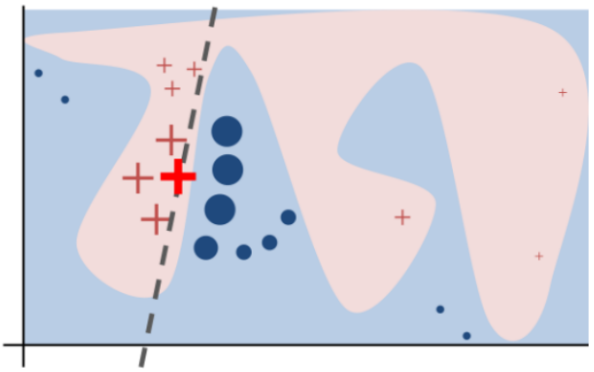

FIGURE 2.8: A summary of the intuition behind LIME, from (Ribeiro et al., 2016). The true decision function of the model is represented by the red/blue background. The bold red circle represents the original instance to be explained, and the other circles and crosses represent perturbed instances. These weighted results are used to train a locally faithful model.

outputs. The best-known example is LIME (Ribeiro et al., 2016), which perturbs the instance to be explained, and uses the output for these perturbed instances to train a new, locally faithful classifier: see Figure 2.8.

In this project, I focused on the 'post hoc - local' category of explanations. This is because these are the explanation techniques are the most commonly used in industry - with SHAP (Lundberg and Lee, 2017) being the most common tool within this category (Bhatt et al., 2020).

Another important consideration is who explanations are for, and for what purpose. (Ribera and Àgata Lapedriza, 2019) provides a useful graphical summary, which is reproduced in Figure 2.9. An example of 'domain experts' would be medical professionals seeking to understand the reasons for an automated diagnosis. A 'lay-user' might be an individual seeking to assert her right to an explanation under GDPR regulations

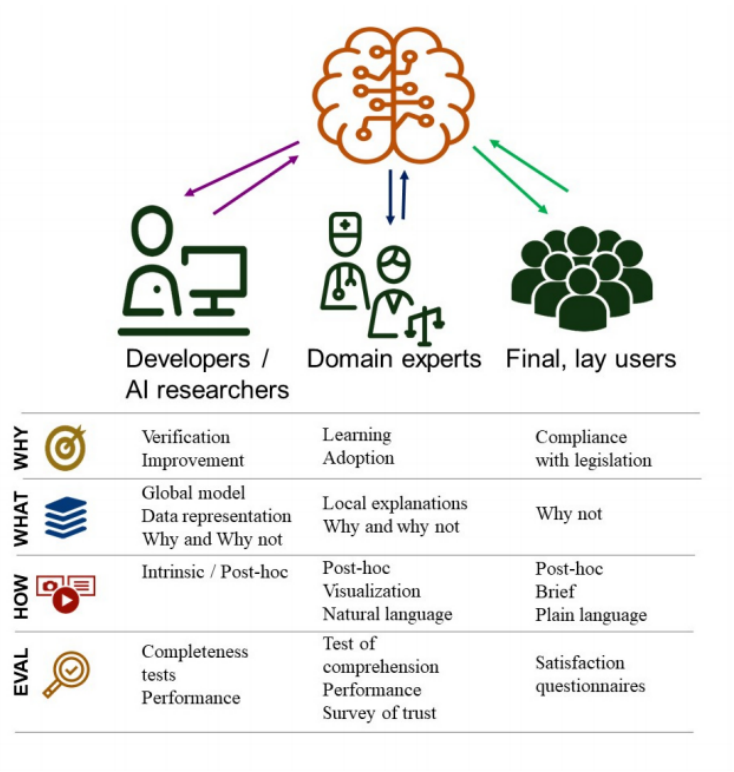

FIGURE 2.9: XAI user groups, reproduced from (Ribera and Àgata Lapedriza, 2019)

(Goodman and Flaxman, 2017). (Bhatt et al., 2020) found that, in practice, the main consumption of explanations is for more prosaic purposes - developers and data scientists attempting to debug, understand and test their models.

2.1.5 Fairness, Bias and Ethics

ML models can be susceptible to bias, if their training data is itself biased. A widely shared example was that of the COMPAS probation system in the USA (Angwin et al., 2016) being biased against black Americans in its decision making.

While the methodology of this study is contested (Flores et al., 2016), it is difficult to argue with the overall notion of bias being an area of concern in ML, with many other examples present in the literature (Caliskan et al., 2017). Clearly, XAI has the potential to improve the situation - a robust and easily applied explanation technique could be routinely deployed to assess ML models for bias (assuming, of course, the consumers of the model agree on what 'bias' means in the specific domain).

One interesting angle borrowed from psychology is the potential for a poor-quality explanation to increase trust in a system, regardless of its accuracy (Rozenblit and Keil, 2002). This is referred to the 'illusion of explanatory depth'. This study noted "*the illusion for explanatory knowledge is most robust where the environment supports real-time explanations with visible mechanisms*" - clearly, XAI techniques fall under this banner. This finding is supported by other studies that consider XAI specifically. For example, (Dzindolet et al., 2003) show explanations can significantly increase trust when they help a user understand why a system may give inaccurate results.

Although further work is required in this area, both studies provide a sense of the importance of providing accurate explanations, from an ethical and practical perspective. In addition, the EU's 'right to an explanation' as part of the GDPR legislation adds further urgency to this work, although the exact interpretation of this right in terms of ML models is still being debated (Goodman and Flaxman, 2017).

Finally, an interesting counter-argument to the proposition that *accurate* explanations are always a universal good is provided by (Samek, 2019). This book chapter persuasively argues that, in certain situations, transparency is not always a desirable for an ML model. Privacy-reduction and allowing the system to be gamed are two such examples. A particularly interesting case is where there is a divergence between audience and beneficiary - with (selectively chosen) explanations being used to influence the audience in a certain direction. For example, the audience of the explanations may be the users of some prediction system in the mould of COMPAS, whereas the benefiary of those explanations would be the developer of the system, assuming they result in greater user satisfaction. Section 2.2.3 gives examples of where explanations are deliberately manipulated, which can be viewed as an extreme example of this line of thought.

2.2 Related Work

This section considers explanation robustness work directly related to this project.

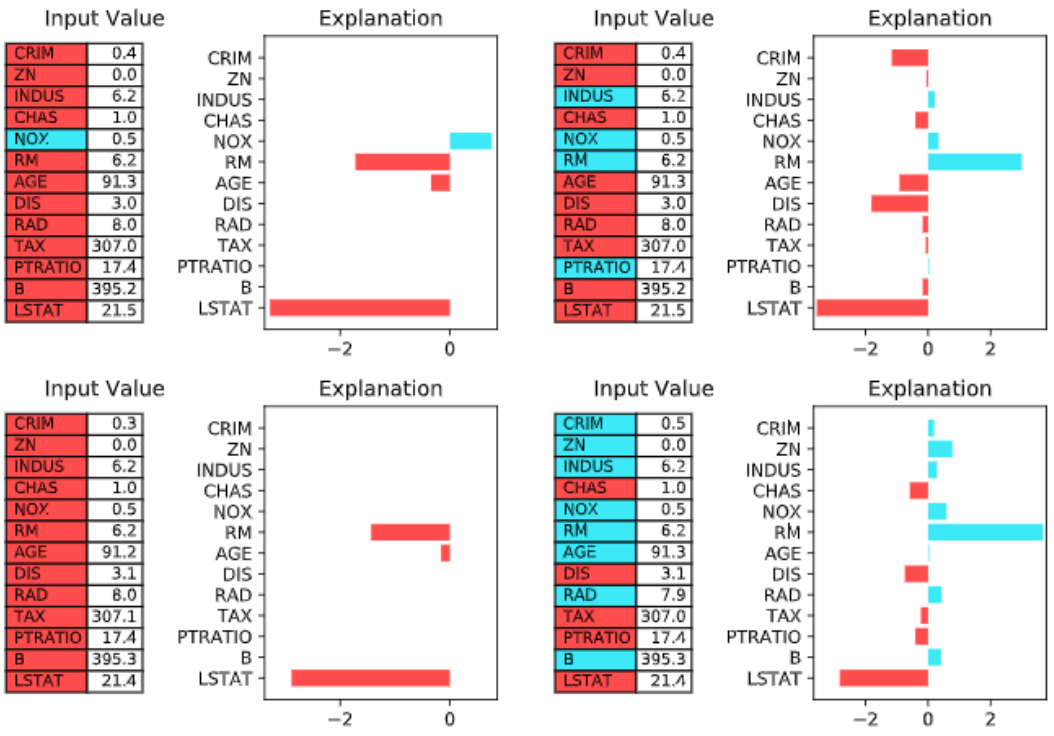

FIGURE 2.10: An example from (Alvarez-Melis and Jaakkola, 2018) showing the impact of very small perturbations on explanations from LIME and SHAP of the BOSTON house price dataset. This implies these explanations suffer from a lack of robustness.

2.2.1 Quantitative measurement of explanation robustness

Section 2.1.4 refers to XAI techniques not being fully reliable, or robust. As this notion of robustness is at the heart of this project, I will formalise what is meant by explanation robustness, and how it can be quantitatively measured.

(Alvarez-Melis and Jaakkola, 2018) provides a useful starting point by defining explanation robustness as inversely proportional to a **local Lipschitz constant**, which measures the deviation in explanation for a small perturbation in the input space. Refer to Figure 2.10 for a concrete example.

This notion of explanation robustness is refined by (Yeh et al., 2019), which divides the metric into two subtly different variants:

- **Sensitivity** - to a small perturbation of the input space. This is essentially equivalent to the local Lipschitz constant. Note that, in theory, the least sensitive possible explanation will be a constant, incorrect one. This is why sensitivity alone does not give a complete picture of robustness.

Definition 1 (Explanation sensitivity)**.** With n respresening the size of the input space, given a black box predictor (model) function $f : \mathbb{R}^n \to \mathbb{R}$, explainer function returning a vector of feature importances$\Phi : \mathbb{R}^n \to \mathbb{R}^n$, max perturbation radius $r : \mathbb{R}$, and $x, x' : \mathbb{R}^n$ representing the original and perturbed instances respectively:

$$S(\Phi, f, x, r) = \max_{||x'-x|| \leq r} ||\Phi(f, x') - \Phi(f, x)||$$

Yeh at al. estimates this metric using Monte Carlo sampling, however it would appear to be an obvious candidate for formal verification methods, given its similarity to the 'epsilon ball' concept referred to in Section 2.1.3.

- **Infidelity** - involves making larger changes to features deemed important by the explanation, and evaluating their effect on the output via mean squared error. A large value implies the explanation is not faithful to the underlying model.

  Definition 2. Given a black box function $f : \mathbb{R}^n \to \mathbb{R}$, explainer function returning a vector of feature importances$\Phi : \mathbb{R}^n \to \mathbb{R}^n$, instance $x : \mathbb{R}$, and $p : \mathbb{R}^n$ representing a meaningful perturbation of interest:

  $$I(\Phi, f, x) = p^T \Phi(f, x) - [f(x) - f(x - p)]^2$$

  A different notation for perturbations is used compared to Definition 1 (p rather than x') as this class of perturbation is different in nature. It represents a larger change to some baseline value, as opposed to a small change within some local radius. It also represents the perturbation itself, rather than the resulting instance.

  Yeh et al. present various ways of generating these 'perturbations of interest' of varying complexity for their infidelity metric. The most simple example would be to take the top n features as given by the explanation, and set these either to zero, or some baseline value. They present an interesting theoretical argument that many existing XAI techniques can be reproduced by using different perturbation approaches, although do not demonstrate this empirically, focusing instead on developing novel techniques.

An interesting section of (Yeh et al., 2019) states:

> "*A natural notion of the goodness of an explanation is to quantify the degree to which it captures how the predictor function itself changes in response to significant perturbations. Along this spirit, [4, 37, 43] propose the completeness axiom for explanations consisting of feature importances, which states that the sum of the feature importances should sum up to the difference in the predictor function value at the given input and some specific baseline.*"

I will refer to this as the completeness 'property' of an explanation technique, as there is no reason why this should be a fundamental axiom of explanations. One could easily write a trivial explainer that violates this property.

'Feature importances' refers to a vector of real numbers giving an indication of how much that feature contributed to the prediction. This is the output of explainer function $\Phi(f, x)$ in the above formulae. Feature importances can be negative - this represents a feature that has made a negative contribution to the prediction, pushing it in a downward direction.

The completeness property is of vital importance to the infidelity metric. Definition 2 takes the square of the difference of two parts:

- '*Expected change*': $p^T \Phi(f, x)$
- '*Actual change*': $f(x) - f(x - p)$

We clearly need these two parts to be measured on the same scale for the output of this metric to be meaningful. For example, consider if perturbation p and explanation output $\Phi(f, x)$ were vectors of floating point values between `[-1, 1]`. But, the model output $f(x)$ was not bounded in this way, and could be between `[-1000, 1000]`, as might be common in a regression problem. In this case, the result of the 'expected change' part of the formula would have little or no bearing on the overall output.

This completeness property is taken as a given in (Yeh et al., 2019), who use SHAP as one of the explanation techniques. However I was unable to find any literature demonstrating this property holding for SHAP. This is investigated further as one of my hypotheses.

2.2.2 Impact of an unstable model on explanation robustness

As noted in Section 2.1.1, ML models are often inherently unstable, with a tendency to learn surface statistical irregularities and a vulnerability to adversarial attacks.

In this context, is it reasonable to expect explanations to exhibit low sensitivity, when the underlying model itself is highly sensitive to input changes? For example, a natural question that arises from Figure 2.10 is - what if the explanation has changed due to a change in the model output? Perhaps the explanation is actually perfectly faithful?

It is challenging to find literature in this area. (Alvarez-Melis and Jaakkola, 2018) briefly mentions this as a challenge, but offers no empirical analysis. Although the infidelity metric proposed by (Yeh et al., 2019) should theoretically be useful in this context (as it measures the explanation's faithfulness to the underlying model), the authors focus on using metrics to optimise other metrics, and do not consider what they tell us about the underlying model.

2.2.3 Adversarial techniques to manipulate explanations

Given we know ML models themselves are vulnerable to adversarial attacks, and explanations exhibit high sensitivity to input changes, it comes as no surprise to find explanations are similarly vulnerable to adversarial attacks.

Some work takes a 'local' approach to manipulating explanations, such as (Dombrowski et al., 2019). The approach is similar to the FGSM concept of using gradient descent to find optimal perturbations to manipulate the explanation. This approach can manipulate various XAI techniques such as integrated gradients and LRP, and give visually arresting results, as shown in Figure 2.11.

Alternatively, a 'global' approach can be taken, which seeks to construct an adversarial classifier which gives incorrect explanations for any instance fed into it. (Slack et al., 2019) is an example of this style of attack, which focusses specifically on manipulating LIME/SHAP (perturbation-based) explanation techniques. It is successful in this, at least in certain datasets, as shown in Figure 2.12. The technique essentially involves detecting whether the perturbations generated by LIME/SHAP (see Figure 2.8), are

FIGURE 2.11: Example of an attack on explanations on a local basis, from (Dombrowski et al., 2019). No modifications were made to the model itself, unlike in Figure 1.3. However, the image itself has been perturbed to generate an explanation that is patently false.

on or off the data manifold[1], as shown in Figure 2.13. Different classifiers can then be deployed in each of these circumstances, allowing explanations to diverge from the true behaviour of the model.

Although the paper produces a series of dramatic demonstrations of biased racist/sexist classifiers going unnoticed, it is highly reliant on the random perturbations generated by LIME/SHAP. An improved sampling approach would be likely to offer a defence. This is demonstrated empirically by (Vreš and Šikonja, 2020), which contributes improvements to LIME/SHAP to keep their sampling on-manifold, and thus reduce the effectiveness of the attack.

Aside from this specific attack on LIME/SHAP, a notable flaw with the global approach in general is that the classifier must be designed with a particular explanation technique in mind. For example, (Heo et al., 2019) focusses on saliency map based interpreters

[1] 'Data manifold' in this context can be understood simply as a cluster of data in n-dimensional space. A two dimensional example is shown in Figure 2.13

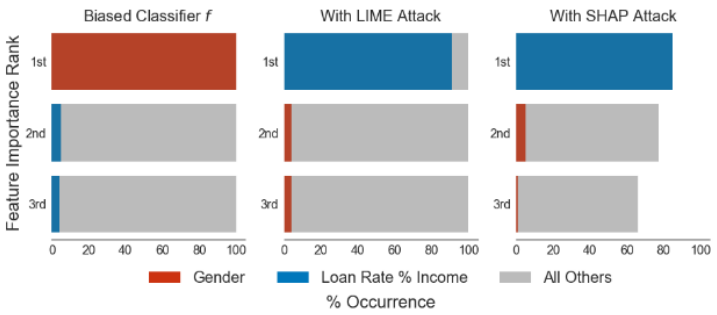

FIGURE 2.12: Example of an attack on explanations on a global basis, from (Slack et al., 2019). The true explanation is shown on the left - gender is the most important feature. However, after adversarial modifications to the model this paper demonstrates, for any single input with similar characteristics to the training set, the explanation incorrectly shows loan rate as the most important feature.

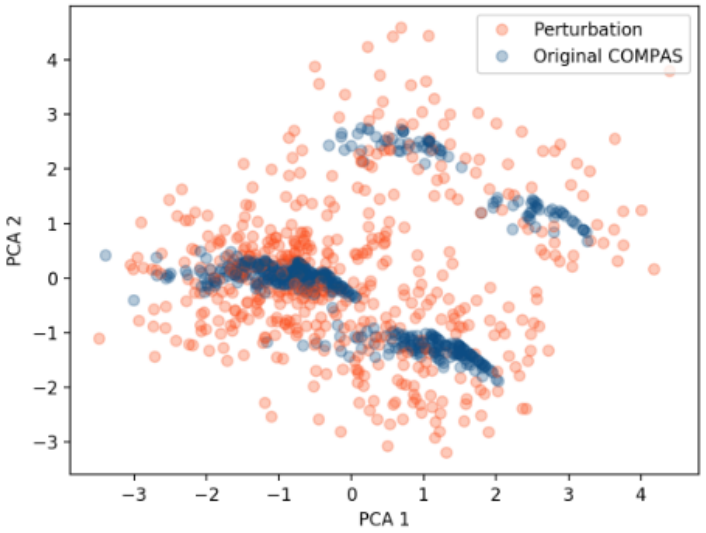

FIGURE 2.13: From (Slack et al., 2019) - PCA applied to the COMPAS dataset, with original points in blue. LIME-style perturbations are in red. It can be easily seen that these perturbations and the original points are separable to a large extent - this is what enables the attack on explanations.

such as LRP, but does not consider linear surrogate models (vice versa for (Slack et al., 2019)).

2.3 Summary

Table 2.1 provides a brief summary of how key papers address my main research questions. It also analyses whether the authors published software that can be used to assess

TABLE 2.1: Summary of related work

Paper	**Robustness in context of underlying model considered?**	**Impact of adversarial attacks considered?**	**Formal verification for XAI considered?**	**Software tool for explanation robustness published?**
On the Robustness of Interpretability Methods (Alvarez-Melis and Jaakkola, 2018)	✗ Sensitivity metric only, which does not consider underlying model.	✗	✗	✗ Code open sourced but as a proof of concept rather than a usable tool.
On the (In)fidelity and Sensitivity of Explanations (Alvarez-Melis and Jaakkola, 2018)	✓ Infidelity measure considers faithfulness of explanation to underlying model, and therefore implicitly its robustness.	✗	✗	✗ Code open sourced but as a proof of concept rather than a usable tool
Fooling LIME and SHAP (Alvarez-Melis and Jaakkola, 2018)	✓ In the context of a specific adversarial attack. More generalisable metrics not proposed.	✓ Impact of attacks on LIME and SHAP considered.	✗	✗
Towards Trustable Explainable AI (Ignatiev, 2020)	✓ Formal verification techniques for XAI implicitly consider model robustness.	✗	✓ Different techniques for a rigorous, logic-based approach to XAI described.	✗

robustness.

Further analysis of each column follows:

- **Robustness in context of underlying model considered?** Definition **??**, of infidelity, should theoretically be able to detect an unstable model. It will be interesting to compare this to the subtly different sensitivity metric (Definition 1), which does not consider the underlying model. Hypotheses 1 and 2 propose to formally investigate this.

- **Impact of adversarial attacks considered?** Various papers, including (Slack et al., 2019), investigate the potential for adversarial attacks on explanations. As far as I am aware, there has not been any work considering the impact of such

attacks on generalisable metrics such as sensitivity and infidelity. This is unsurprising, given there is no agreed standard for measuring robustness of explanations. It will be interesting to assess the impact of these adversarial attacks, and whether generalisable robustness metrics are sufficient to detect one is taking place. Hypothesis 3 focuses on this.

- **Formal verification for XAI considered?** (Ignatiev, 2020) summarises prior work on rigorous, formal methods for producing explanations. It is unrealistic to think I will be able to improve on such methods in this project. I focused my work in this area towards combining heuristic and formal methods in a manner that may not be logically rigorous, but still gives useful data on robustness of explanations, perhaps for a subset of important features.

- **Software tool for explanation robustness published?** Although some papers have helpfully published code relating to their work, this is not in a form that could easily be used by a generalist ML practitioner to quickly perform a sanity check on a post-hoc explanation. As far as I am aware, no such tool exists.

Chapter 3

Requirements and Methodology

This section has changed slightly from the original Research Report for this project. A summary of the changes is given in Appendix C.

3.1 Deliverables

This project involves three separate work products:

1. Research Report and literature review (submitted April 2021).
2. MSc Thesis Report (this document).
3. Software package to assess robustness of post-hoc explanations. This meets Objective 7 as listed in Section 1.3.

3.2 Research Hypotheses

My first hypothesis was that explanation sensitivity and infidelity metrics can be implemented in a way that improves the existing implementation (Yeh et al., 2019). The existing implementation by the authors of this paper was not intended for wider use, but rather as a means of replicating the results in this paper. Refer to Table 3.1 for a comparison. My implementation is novel, and starts from scratch - it does not build on the existing implementation (aside from using the same underlying definitions).

TABLE 3.1: Comparison of existing and planned implementation of explanation metrics

Existing implementation (Yeh et al., 2019)	**Planned implementation**
Tight coupling to specific models and explanation techniques used in the paper.	Model and explanation technique-agnostic. Can be used with any scikit-like model (that provides a predict() function), and any explanation (list of feature importances).
Cannot handle nominal data. Requires pre-processing to convert to numerical format (such as one-hot encoding).	Understands nominal input, both ordered and unordered (ordinal). Allows instances to be provided without pre-processing, with this baked into the model itself, as is often the case in real-life deployments of ML models.
No documentation on how to use the metrics.	Full documentation provided, including a worked example. Code commented where required.
Code in GitHub only.	Code published to PyPI (PYpI, 2021), a popular repository of Python packages.
No unit tests or systematic logging. Limited error handling.	Multi-level logging allows internal working to be inspected. Input validation and error handling provided. Unit tests provide a high level of coverage.

Hypothesis 1. *The explanation robustness metrics sensitivity and infidelity can be implemented in a way that is more aligned with engineering best practices, allows application to nominal datasets and arbitrary models/explanations, and is published on a widely used package repository for wider use.*

This hypothesis includes validating the metrics give reasonable results. The 'completeness' property validation of explanations (as discussed in Section 2.2.1 was of sufficient interest to separate out into its own hypothesis:

Hypothesis 2. *The completeness property for explanations - that a change in explanation feature importances will always sum to the corresponding change in model output - holds for SHAP, but is not a fundamental property of all explanation techniques.*

The next stage of my work was to apply these sensitivity and infidelity metrics to real-life nominal datasets. Firstly, the metrics were applied to 'adversarial classifiers', as described in Section 2.2.3. The explanations generated by these classifiers are clearly not robust, as they are designed specifically to give a misleading explanation. I did not believe, however, the metrics would be sophisticated enough to detect this:

Hypothesis 3. *Explanation robustness measures such as sensitivity and infidelity incorrectly measure explanations of 'adversarial classifier' models as having high robustness, when in fact the opposite is true.*

My next hypothesis is similar, but considered the situation of a sub-optimal model - one that has been overfitted to training data. Therefore, the model itself (as opposed to the explanation) is lacking robustness:

Hypothesis 4. *Explanation robustness measures such as sensitivity and infidelity incorrectly measure explanations of overfitted models as having low robustness, even when the explanations are in fact consistent with the model.*

3.3 Requirements Analysis

The requirements for this project divide into two sections: experimental requirements (for research hypotheses), and software requirements (for Objective 7). Software requirements subdivide further into functional and non-functional requirements.

These requirements expand on and formalise the list of objectives in Section 1.3. A reference to one or more objectives is provided for each requirement. The tables of requirements were used to produce a project plan in Appendix B.

Tables presented in this section include a *Completed?* column, which summarises whether the requirement was met. The majority of requirements have been met, including all high-priority items. Some medium/low priority requirements have not been met due to time constraints. I found the novel implementation of existing explanation robustness metrics was both more complex and more interesting than anticipated. This prevented me from assessing other metrics.

3.3.1 Experimental Requirements

Refer to Table 3.2 and Table 3.3.

TABLE 3.2: Experimental requirements (part 1). 'Obj' refers to the objectives in Section 1.3

Ref	Name	Description	Obj.	Pri.	Completed?
E2	Select datasets for classification problem	Identify two suitable nominal datasets for a supervised **classification** model	1	High	✓
E3	Train 'baseline' model	Use TensorFlow to train a high-performing model. Classification performance should be comparable to state of the art for the chosen datasets for accuracy and F-measure.	2	High	✓
E4	Train 'adversarial' model	Use TensorFlow and the techniques described in (Slack et al., 2019) to train an adversarial model, achieving similar results (no material loss in accuracy, and a significant change in the explanation).	2	High	✓ Was also extended to a non-trivial MLP based adversarial model.
E5	Train 'sub-optimal' model	Use TensorFlow to train an over-fitted model that does not generalise well to unseen data. Should achieve high accuracy on the training set, but materially lower accuracy on an unseen test set.	2	Med.	✓
E6	Generate primary explanations	Use SHAP to generate explanations for a sample of three randomly chosen input instances for each model. Provide visualisations of explanations. Ensure explanations pass a 'sanity check' of accuracy.	3	High	✓
E15	Assess completeness property of explanations	Demonstrate completeness property holds for SHAP, but not for all explanation techniques.	5	Low	✓
E16	Implement sensitivity and infidelity metrics	Implement metrics in a way that is model-agnostic, and handles categorical as well as numerical features. Verify the metrics are giving reasonable results using toy models and Iris dataset.	4	High	✓
E8	Assess explanations against sensitivity metric	Metric as defined in (Yeh et al., 2019). Aim to disprove null hypothesis for 3 and 4.	6	High	✓
E9	Assess explanations against infidelity metric	Metric as defined in (Yeh et al., 2019). Aim to disprove null hypothesis for 3 and 4.	6	High	✓
E10	Assess explanations against other metrics	Other metrics defined in the literature may prove useful in evaluating hypotheses. Aim to disprove null hypothesis for 3 and 4.	4	Low	✗ Not possible due to time constraints.

TABLE 3.3: Experimental requirements (part 2). 'Obj' refers to the objectives in Section 1.3

Ref	Name	Description	Obj.	Pri.	Completed?
E13	Present methodology and findings	Use a Jupyter notebook and host on GitHub. Results should be replicated using the notebook, which should also provide visualisations and brief explanations for why certain results are seen.	All	Med.	✓
E14	Overall conclusions	Assess the the evaluation against each hypothesis - form conclusions about the overall experiment, and how this alters my views on the preexisting literature.	All	High	✓

3.3.2 Software Requirements

All these requirements are linked to Objective 7 as described in Section 1.3. Refer to Table 3.4 and Table 3.5 for functional and non-functional requirements respectively.

TABLE 3.4: Functional software requirements

Ref.	Name	Description	Pri.	Completed?
F1	Generate sensitivity and infidelity statistics	Take in a model, a list of feature importances (i.e. an explanation) and an instance. Generate statistics for sensitivity and infidelity for that explanation. For nominal data types only.	High	✓
F2	Generate other robustness statistics	As for F1, but using other metrics, either from the wider literature, or created as part of this project.	Med.	✗ Not possible due to time constraints.
F3	Allow more sophisticated input of feature importances	Allow the input of feature importances (in F1/F2) to be given as a score for each feature, as opposed to a simple list of features.	Low	✓
F4	Compatibility with any scikit classifier	The model input should take in any scikit-learn (Python) model. (Note this allows TensorFlow/Keras models via the KerasClassifier class).	High	✓
F5	Show progress	A progress bar or similar shows how long the user will need to wait for results.	Low	✗ Not possible due to time constraints.
F6	Error handling	Errors are handled elegantly, with the user given clear instructions on how to fix any issues with their input.	Med.	✓
F7	Logging	Logs are piped to *stdout* which give a summary of what the application is doing internally.	Med.	✓
F8	Extend to images	Extend functionality to image classification problems, with appropriate visualisations.	Low	✓ Not specifically tested as part of this project, but metrics are datatype-agnostic.

3.4 Planned tooling and architecture

This project was implemented as a Python codebase. It is hosted on my public GitHub at this URL: `https://github.com/pidg3/hw-dissertation`. A high-level summary of the intended architecture for the published software is shown in Figure 3.1. This has changed little during implementation.

Tools to be used include:

TABLE 3.5: Non-functional software requirements

Ref.	Name	Description	Pri.	Completed?
N1	Documentation and usability	Clear documentation should be written, describing available functionality, limitations, and a ''how-to' guide.	High	✓
N2	Performance	The software should be able to produce robustness statistics within a reasonable timeframe (i.e. under 60 seconds) for a Keras model.	Med.	Partial - met for infidelity metric. Not met for sensitivity metric.
N3	Testing	Unit tests and end to end automated tests should be written.	Med.	✓
N4	Extensibility	Software should be written to allow addition of extra robustness statistics.	Low	✓
N5	Licensing	Use of GNU General Public License v3.0 should be document in the project README. Refer to Chapter 4 for further details.	High	✓

- TensorFlow (Abadi et al., 2016): training and running of ML models, focused largely on ANNs.
- Keras (Chollet et al., 2015): API for training DNNs, built on top of TensorFlow (and now integrated into the platform).
- Scikit-Learn: Python package providing a number of ML tools and techniques, as well as a simple API for training and running models. This API has become the defacto standard for ML work.
- SHAP (Lundberg and Lee, 2017): post-hoc explainabilty tool that will form the baseline for my investigations.

3.5 Success criteria

All experimental hypotheses have a control group. Hypothesis 2 will use an alternative explanation method as the control, with the aim of demonstrating differences in completeness properties. For hypotheses 3 and 4, this is the 'baseline' high performing model. Robustness of this model will be contrasted with the 'adversarial' and 'sub-optimal' models.

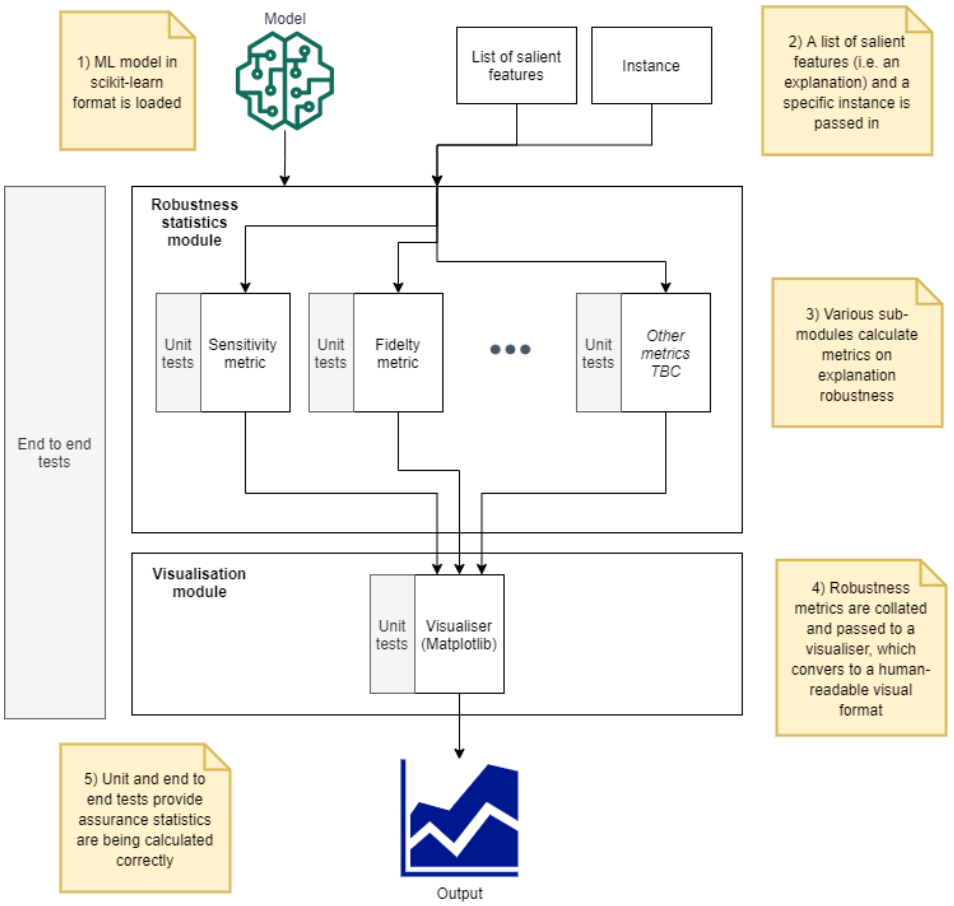

FIGURE 3.1: A high-level informal summary of the intended application architecture. Additional information given in the yellow stickers.

I will view Hypothesis 1 as a success if it implements all 'Medium' and 'High' priority requirements given in Table 3.4 and Table 3.5, the metrics can be validated successfully using toy datasets, and open-source software is published.

I will carry out an evaluation against each objective in Section 1.3 as the final part of this project.

Chapter 4

Implementation

4.1 Sensitivity and infidelity metrics

This section describes the algorithms used, and how they were improved over time. My work has been published as a PyPI package (PYpI, 2021) - see `https://pypi.org/project/exrt/`. For information on the tooling, deployment and testing patterns, refer to Section 4.2.

The original versions of these metrics were defined by (Yeh et al., 2019), who provide an implementation that can be found at `https://github.com/chihkuanyeh/saliency_evaluation`. My implementation is completely separate, and contains several novel features and improvements as summarised in Table 3.1.

4.1.1 Sensitivity

This metric attempts to find the maximum difference in explanation caused by a 'small' perturbation to the input. Refer to Definition 1 for the formal definition, and Figure 4.1 for a simplified graphical summary.

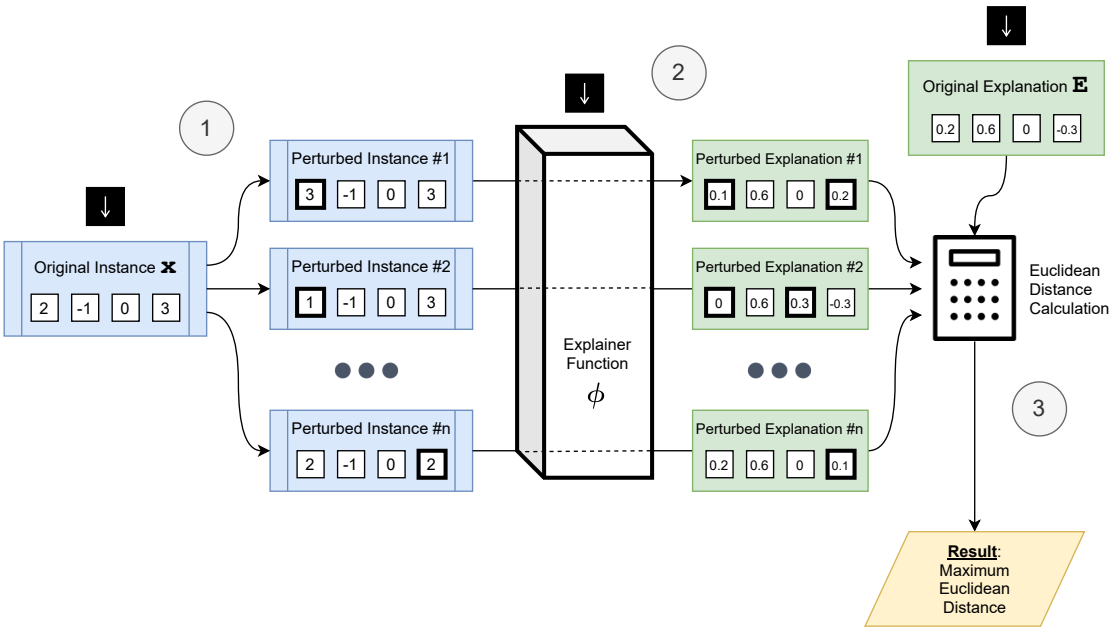

FIGURE 4.1: Simplified graphical summary of the sensitivity metric implementation. Black squares with arrows show inputs. (1) shows the original instance being perturbed multiple times. (2) shows an explanation being calculated for each perturbed instance.

The obvious question for an implementation is - how to define a 'small' perturbation when we have a mix of numerical and categorical[1] features, and therefore cannot simply measure euclidean distance? My implementation made the following initial assumptions:

- A small proportion of features should be perturbed, default set as 10% (rounded up).
- For numerical features, perturb by a small percentage in both directions, default set as 10%.
- For nominal features, perturb by choosing a different value randomly.
- For ordinal features, perturb by shifting one place in both directions (for example, a month feature with value `May` would be perturbed to both `April` or `June`.

Initial attempts at validation showed the metric was not deterministic. This was because categorical features were being perturbed randomly to a single value, so the whole input

[1]The following definitions of non-numerical features are used in this document:

- Categorical - any non-numerical feature.
- Nominal - an unordered non-numerical feature, such as 'weather'.
- Ordinal - an ordered non-numerical feature, such as 'month'.

space was not being explored. The algorithm was adjusted to test all possible values (at the expense of running time).

Another issue was numerical inputs with zero value. These were not being perturbed, as the adjustment was based on a percentage. After experimenting with adding arbitrary constants, I instead settled on the approach of basing the perturbation value on a baseline value for that feature, calculated from training data before running the algorithm.

A summary of the final algorithm is summarised in Algorithm 1.

The result is bounded as follows:

- Minimum value: 0
- Maximum value: the range of possible model outputs (for example in a binary classification problem, maximum value would be 1).

4.1.2 Infidelity

This metric attempts to calculate an explanation's degree of unfaithfulness to the underlying model. As with sensitivity, a high numerical output suggests a lack of robustness.

It perturbs the features the explanation tells it are most important, and checks whether the model output shows a large difference, as would be expected if the features are truly important. Refer to Definition 2 for the formal definition, and Figure 4.2 for a simplified graphical summary.

Infidelity addresses an obvious issue with sensitivity - the situation where the underlying model is unstable, and so a small change in model input causes the prediction to vary significantly, as well as the explanation of that prediction. In this case, sensitivity would suggest the explanation is not robust, which may be incorrect.

Features are perturbed to a baseline value. This must be calculated in advance of the calculation being run, and provided as a metadata JSON file. The method of calculating the baseline value varies between data type:

- For numerical features - the mean.

Algorithm 1: Simplified summary of sensitivity metric implementation

```
Input : φ ← explainer function for a particular model
Input : E ← original explanation
Input : x ← instance
Input : n ← number of features to perturb
x_copy ← x ;
for each n do
    max_difference ← 0 ;
    for each feature in x_copy do
        for each possible perturbation do
            /* 'Possible perturbations' vary depending on feature
               data type                                          */
            x_p ← perturbation applied to x_copy ;
            /* Calculate an explanation, and the difference between
               it and the original explanation                    */
            E_p ← φ(x_p) ;
            this_difference ← (E − E_p) ;
            if this_difference > max_difference then
                max_difference ← this_difference ;
                x_max_perturbation ← x_p ;
            end
        end
    end
    x_copy ← x_max_perturbation ;
end
/* x_copy is now the perturbation causing maximum explanation
   difference                                                    */
E_p ← φ(x_copy) ;
return L2Norm(E_p − E) ;
```

- For nominal (unordered) features - the mode.
- For ordinal - the median value for ordered set.

The main challenge in implementing this metric for non-numerical features was how to calculate the value of a perturbation. While this is trivial for numerical features, it is not obvious what value the perturbation of, say a `marital_status` feature from `married` to `single` should take. The values are calculated as follows:

- For ordinal features - difference in position, divided by total number of possible values for the feature.

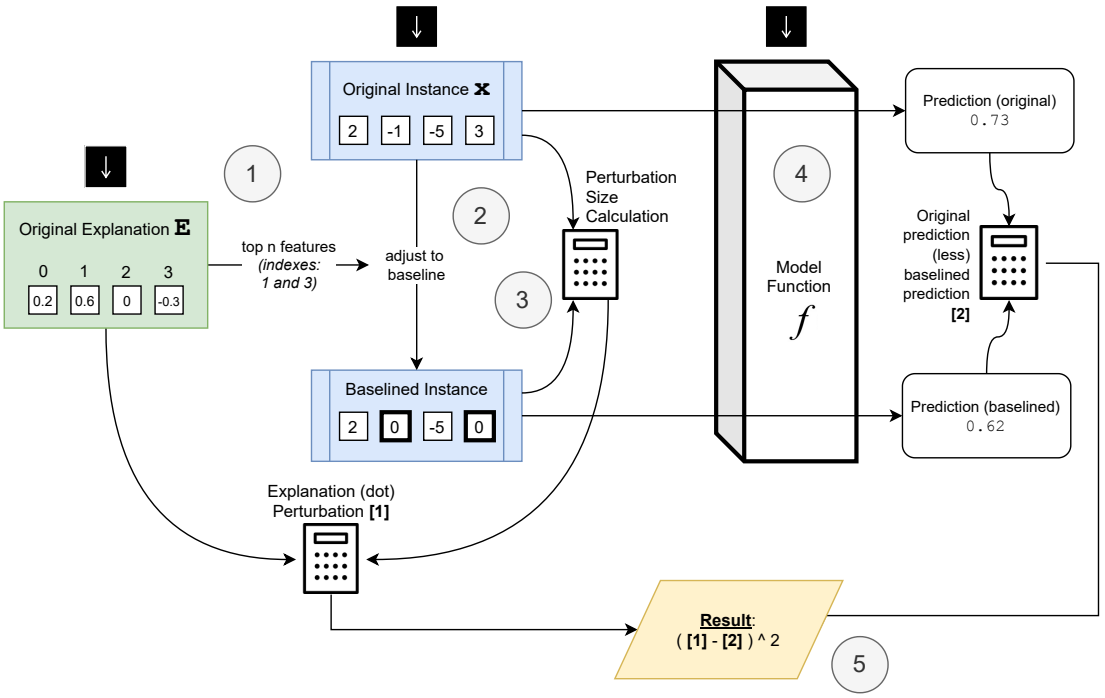

FIGURE 4.2: Simplified graphical summary of the infidelity metric implementation. Black squares with arrows show inputs. (1) shows most important features being identified from the original explanation. (2) shows these features being adjusted to some baseline value. (3) shows a perturbation value being calculated by comparing the two instances (non-trivial for categorical data). (4) shows the two instances being input to a predictor model function. (5) shows the mathematical calculation as given in Definition 2.

- For nominal (unordered) features - before the main part of the algorithm is run, calculate an ordering of the values by substituting each possible value. Then apply the same approach as for ordinal features.

Initially, results from validating this algorithm (see Section 5.2) were highly inconsistent. The issue was feature explanation importances being able to take positive or negative values. The algorithm used absolute values to calculate a subset of features with the highest importances. However, I was not accounting for the fact the two perturbations with opposite signs could cancel out each others' impact on the model prediction. The algorithm was rewritten to calculate the impact of one perturbation at a time, and take a running total of infidelity from each perturbation.

A summary of the final algorithm is summarised in Algorithm 2.

The result is bounded as follows:

- Minimum value: 0

- Maximum value: no theoretical maximum

Algorithm 2: Simplified summary of infidelity metric implementation

```
Input  : E ← original explanation
Input  : x ← instance
Input  : f ← model (predictor)
[e] ← most important features (absolute terms) ← E ;
cumulative_infidelity ← 0 ;
for each e do
    /* Instance changed to some baseline value, and perturbation
       size calculated                                          */
    x' ← perturb_instance(x, e) ;
    p ← calculate_perturbation(x, x') ;
    expected_difference = dot_product(p, E) ;
    actual_difference = f(x) − f(x') ;
    /* Apply the function as per formal definition to calculate
       infidelity for this feature                              */
    infidelity = square(expected_difference - actual_difference) ;
    /* Add to a running total                                   */
    cumulative_infidelity += infidelity ;
end
return cumulative_infidelity ;
```

4.2 Open source publication

My work on these metrics has been published as a Python package called `exrt` - EXplainability Robustness Toolbox. This has been uploaded to the PyPI repository of Python packages for others to easily use, see Figure 4.3. It can be found at the following URL: https://pypi.org/project/exrt/.

I have adopted professional software engineering practices for writing and publishing this package, including:

- High-quality documentation, including a full worked example.
- Clearly stating the licence used (GPL).
- Strong unit testing coverage, as well as end to end automated tests.

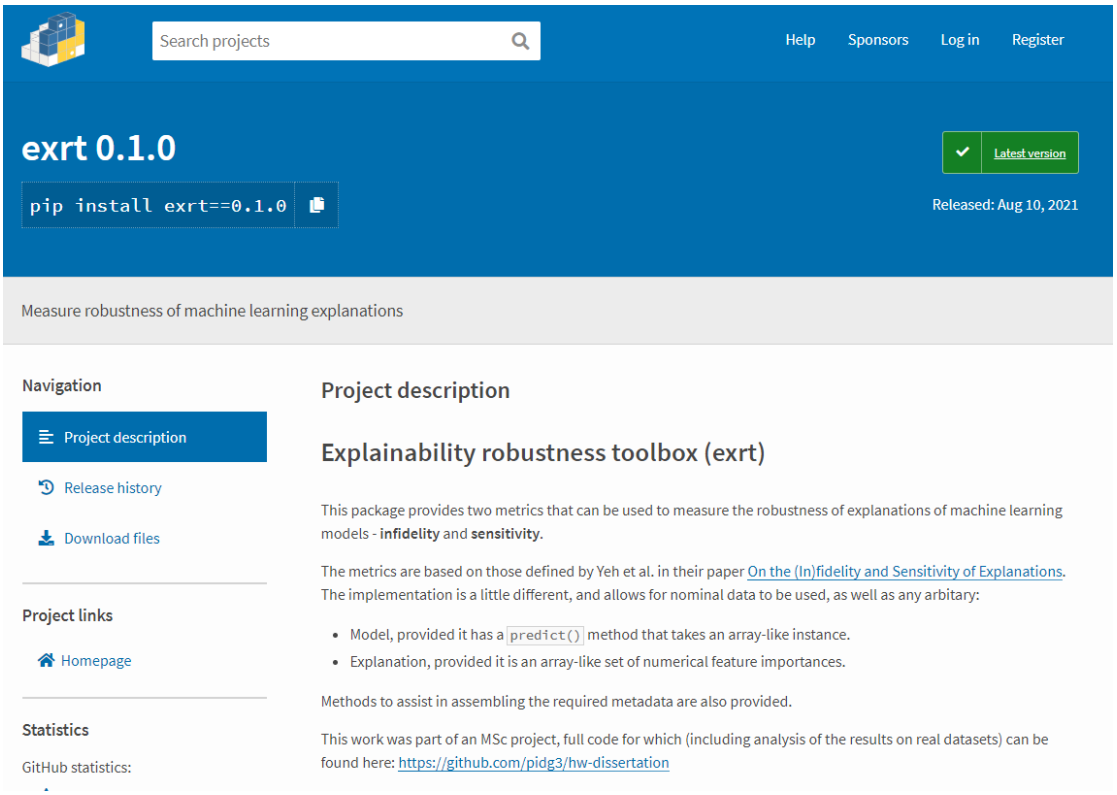

FIGURE 4.3: Screenshot showing `exrt` published on PyPI package repository

Based on searching the PyPI repository as well as my literature review, I believe this is the first tool of its kind published for open-source use. While there are some limitations for its use (see chapter 6), this contribution should prove useful for those seeking a level of assurance their post-hoc machine learning explanations are robust.

4.3 Machine learning models and explanations

I chose two different datasets with contrasting characteristics. Both are well-known datasets obtained from the UCI Machine Learning Repository (Dua and Graff, 2017).

- 'Bank Marketing' (Moro et al., 2014) - a binary classification problem, containing metadata on telephone calls between bank operatives and customers, and using this to predict the customers' likelihood of opening a current account. Features are a mix of numerical, nominal and ordinal. Output classes are unbalanced, with 88% of instances representing customers who did not take out a loan.
- 'Spambase'[2] - a binary classification problem, using information within an email

[2]No reference required as per UCI guidance for this dataset

to predict whether or not it is a spam message. This is not an natural language processing problem, but rather one where the text has been analysed in advance, and data presented as numerical columns only. Output classes are more balanced, and split 40% positive (spam) and 60% negative (not spam).

Multi-layer perceptron (MLP) models for both datasets were built using TensorFlow (Abadi et al., 2016). Through trial and error, I found three layers of 128 nodes, all with ReLU activation functions and 20% dropout, gave strong results.

The models were built using the 'feature column' functionality of TensorFlow. This means all pre-processing is built in to the models themselves, rather than being an additional, separate step that must be completed. This is by design - I wanted the metrics to use the same, raw, inputs as the models. I felt this would make the metrics simpler to use.

SHAP (Lundberg and Lee, 2017) was successfully used to generate explanations for these models. SHAP provides an explanation in the form of 'feature importances' - a list of numbers, with each number representing the size of its contribution to the prediction (which can be negative - i.e., the predicted value was given in spite of, rather than because of, that feature). SHAP provides a number of visualisation tools - for consistency in this report, I will use the `summary_plot` style, refer to Figure 4.4 for an example.

Two models trained for specific purposes are used to assess the explanation metrics. These are detailed below.

4.3.1 Adversarial models

These models are required in order to test Hypothesis 3. I used the Spambase dataset for this model type. As discussed in Section 2.2.3, an adversarial model in this context is one that is designed to give a misleading explanation for any input instance.

In this project, I have extended the work by (Slack et al., 2019). In this paper, the authors demonstrate a technique to teach a model to determine whether it is being asked for a prediction, or an explanation. The model then invokes one of two possible 'sub-models', which can give completely different results. They demonstrate this using

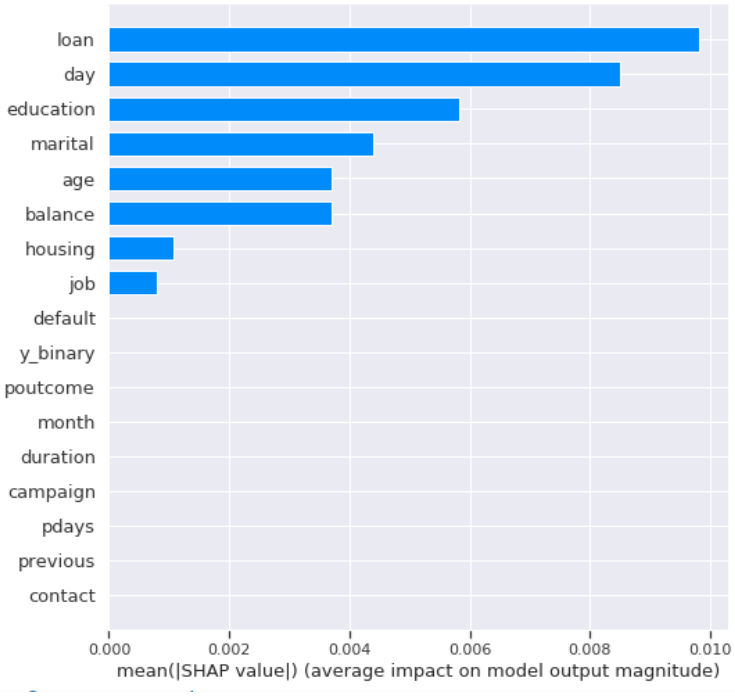

FIGURE 4.4: Example SHAP explanation visualisation, showing feature importances for the 'Bank Marketing' dataset. This shows the most important features to this prediction are loan (whether customer previously took out a loan), day [of the week] and education [level].

trivial models - those that only examine a single feature, and assess whether it is above a constant value.

Reusing code open-sourced by the authors, I implemented the adversarial model concept on the Spambase dataset. I successfully extended their work to use a non-trivial MLP model as the 'explanation' model, while keeping a trivial 'prediction' model which only examined a single feature. The different models involved are summarised in Figure 4.5.

In itself, this is an interesting finding. I was unsure whether it would be possible to extend this technique to include MLP models used as the explanation facade. This finding increases the possibility of this approach being a threat in practice, as the 'fake explanation' appears vastly more realistic when it is generated by an MLP model. Figures Figure 4.6, Figure 4.7 and Figure 4.8 demonstrate this. Note that all these models give the same result when asked for a prediction - they use the value of `capital_run_length_longest` only.

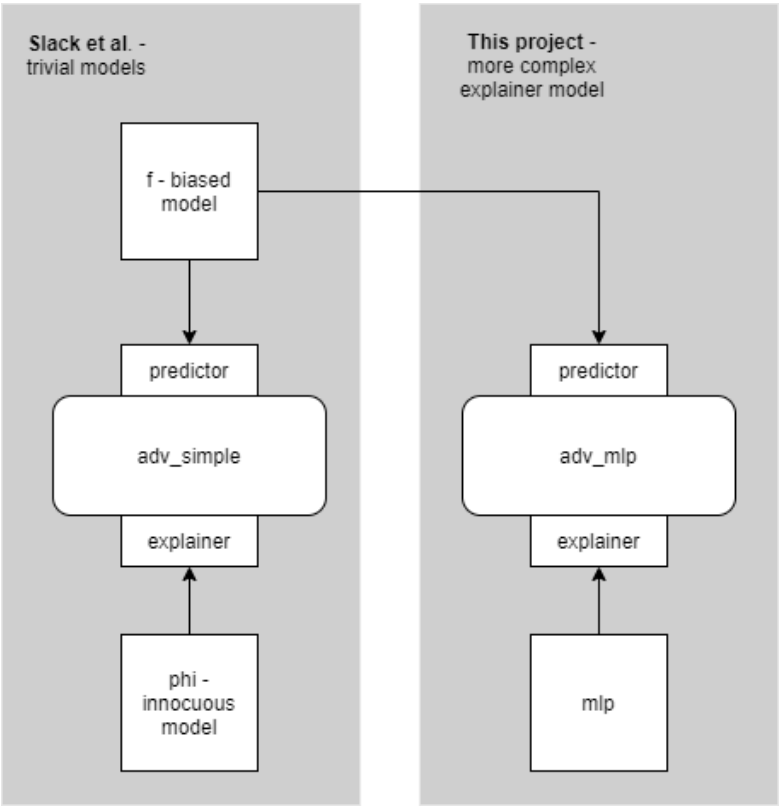

FIGURE 4.5: Adversarial model summary. f and ϕ (phi) are both trivial models which use a single feature value. This project keeps f as the biased model to be obscured, but used an mlp model as the explanation facade. This means a more realistic explanation is generated, albeit one that still bears no relation to the real prediction.

4.3.2 Overfitted models

These models are required in order to test Hypothesis 4. I used the Bank Marketing dataset for this model type. As this dataset is unbalanced, accuracy is an unsuitable training metric - the model can achieve high (88%) accuracy, simply by assigning a negative class to all instances. Area under curve (AUC) is used instead - this is a more sophisticated metric that uses false and true positive/negative rates to give a probability of the model ranking an arbitrary positive instance more highly than an arbitrary negative instance.

Suitable hyperparameters for training overfitted models were found by testing different numbers of training epochs, and toggling dropout on and off. Dropout (Hinton et al., 2012) is a technique used to reduce overfitting during neural network training - a certain percentage of connections between neurons is randomly removed in each training run, and thus the network is trained to avoid over-reliance on a particular set of connections.

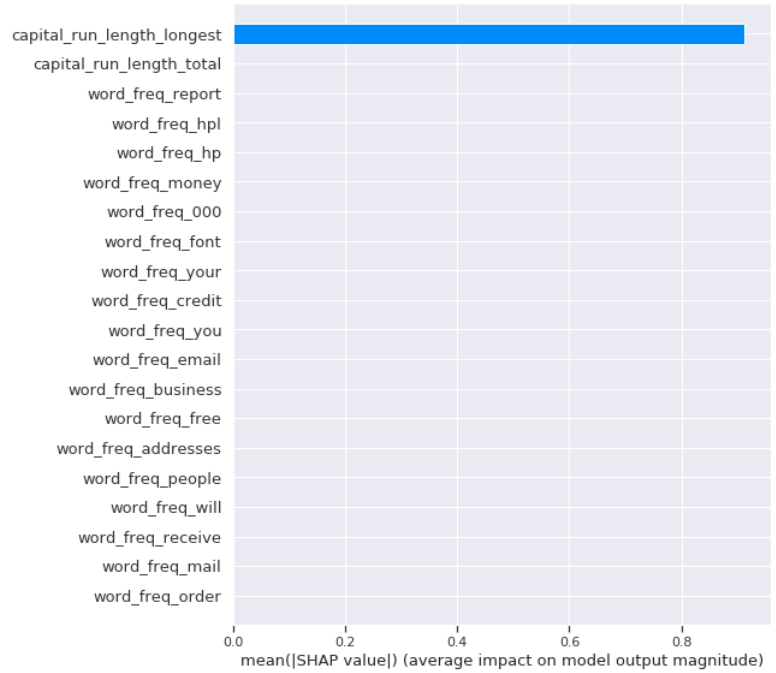

FIGURE 4.6: Explanation of biased model f. This is the 'true' explanation, correctly showing `capital_run_length` as the feature used to generate the prediction.

The results are shown in Figure 4.9. This shows 40 epochs with no dropout exhibits overfitted behaviour - high training set accuracy, but low test set accuracy. These hyperparameters are used for the overfitted bank marketing model.

The baseline (control) model used 10 epochs, with dropout - Figure 4.9 shows this to have optimal test AUC.

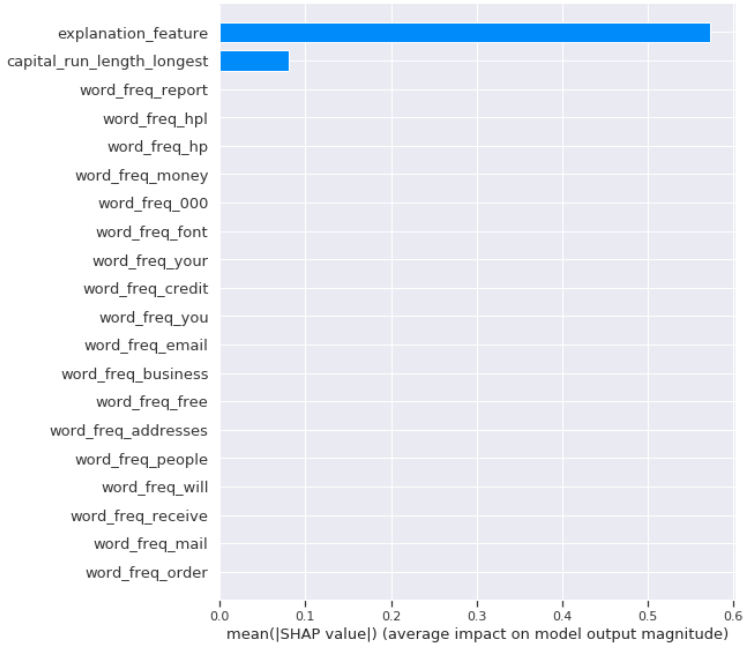

FIGURE 4.7: Explanation of 'adv_simple' model - as used in (Slack et al., 2019). Here we can see a false explanation, suggesting the prediction is almost entirely due to some false 'explanation feature'. This would be unusual in practice for an MLP model.

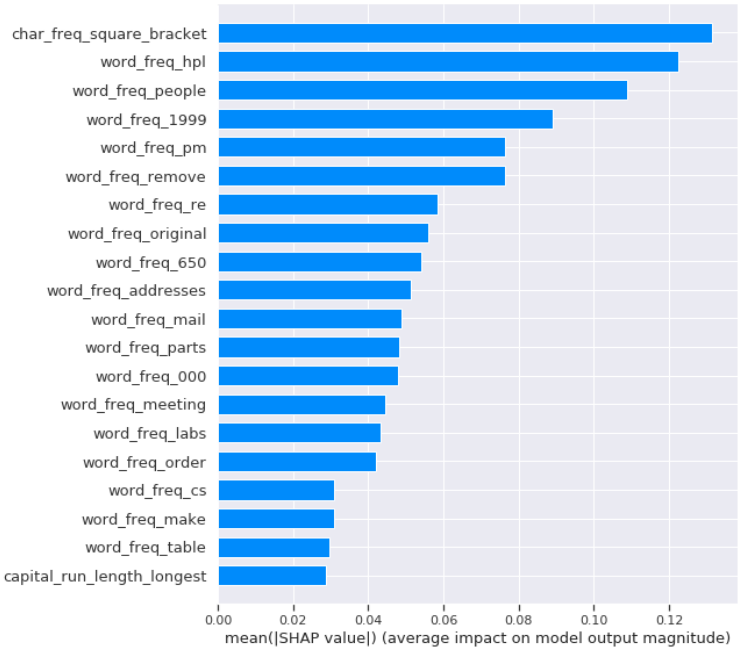

FIGURE 4.8: Explanation of 'adv_mlp' model - while this still gives the same prediction as the other models, it provides a false explanation that is far more realstic, with multiple features showing they are relevant to the prediction.

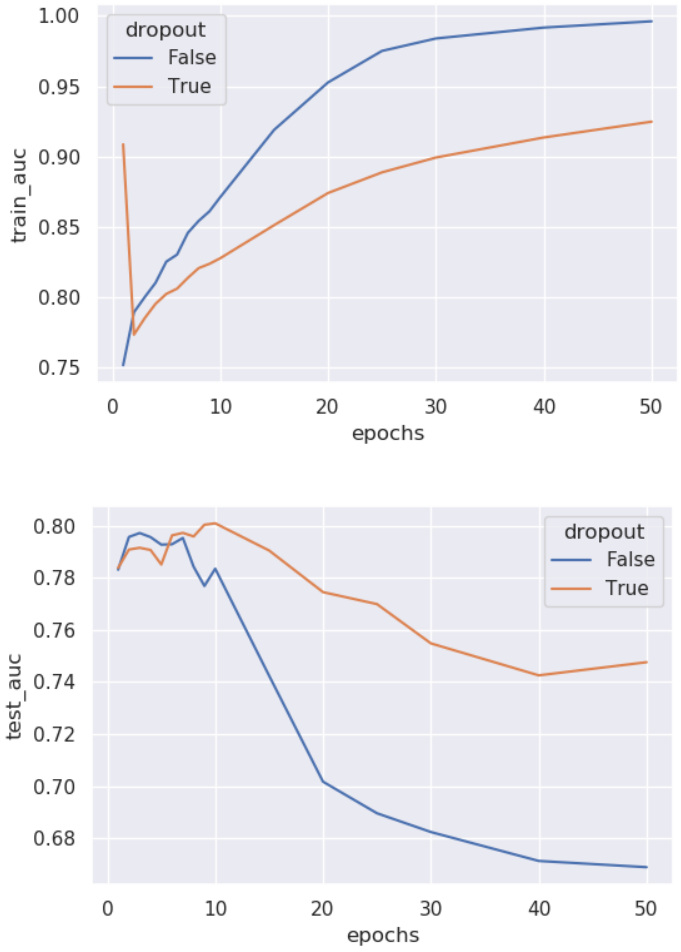

FIGURE 4.9: Results of an exercise to determine how to train an overfitted model for bank marketing dataset. The aim was to produce a model with a large difference between training and test set accuracy, which is achieved around 40 epochs (with diminishing returns beyond that) and no dropout.

Chapter 5

Evaluation

Code for all experiments is hosted on GitHub: https://github.com/pidg3/hw-dissertation/. Full results can be found within these notebooks.

5.1 Evaluation strategy

The aim of this chapter is to evaluate the robustness metrics described in Chapter 4, and thus provide results for each of my four hypotheses. The approach is split into two parts:

1. Building confidence in the metrics by a series of simple experiments where we know the expected results (Hypothesis 1):
    - Apply the metrics to a series of simple linear models - those that assume a linear relationship between one or more input variables (x_n) and a single output variable y. Due to these models' inherent simplicity and interpretability, we can easily determine the expected results and compare to the actual results.
    - Use the Iris (Fisher, 1936) dataset, and a simple MLP model for experimentation on a real-life problem. For sensitivity, compare the metric result with 'uncertainty' - the degree to which the predictor model is unsure of the correct prediction. Higher uncertainty should equate to higher sensitivity. For infidelity, progressively add noise to the explanation. As this reduces its faithfulness to the model, infidelity should progressively increase.

- For Hypothesis 2 specifically - attempt to validate the completeness property (as defined in Section 2.2.1 of SHAP and LIME by comparing a number of instances' prediction and sum of explanation values. Add noise to the instances in order to take them significantly out of distribution and assess whether the property still holds.

2. Deploying the metrics to non-trivial models/explanations, using the 'Bank Marketing' and 'Spambase' models as described in Section 4.3. Attempt to validate Hypotheses 3 and 4; whether the metrics are able to detect overfitted and adversarial models respectively.

Further detail on each of these experiments, as well as the results, is provided in the remainder of this chapter.

5.2 Metric validation

This section aims to build confidence in the metrics, and thus validate Hypothesis 1. All the results shown here are using the 'final' algorithms, after the improvements documented in Section 4.1 were implemented.

5.2.1 Linear models

Firstly, for fidelity, I used a model with the form $y = w_1x_1 + w_2x_2$, where the w values are weights, and the x values inputs. Values were set as $x_1 = 2$, $x_2 = 6$, $w_1 = 1$, $w_2 = 0$. Baseline values (used by both metrics to calculate perturbation values) for the two feature inputs were set as `0.5`.

Explanation importance values for the two features are represented as e_1 and e_2. I conducted three experiments, which are described in captions for Figure 5.1, Figure 5.2 and Figure 5.3. In all cases, the results were in line with expectations.

For sensitivity, the approach used a different model. A stepwise linear function was defined, as summarised in Algorithm 3. Discontinuities are set at $x_0 = 0$ and $x_0 = 2$. Inputs were then varied between `[-4, 4]` for both x_0 and x_1. Results are in line with

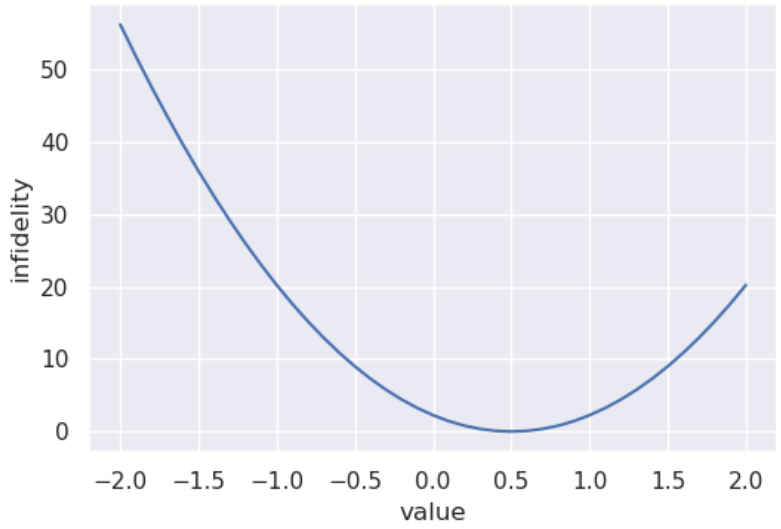

FIGURE 5.1: Set $e_2 = 0$, and vary e_1 between `[-2, 2]`. This was expected to show an exponential increase in infidelity, with a minima at $e_1 = 0.5$ (the feature baseline value). This result is in line with the expectation.

.

expectations, showing higher sensitivity values for x_0 around the discontinuities - refer to Figure 5.4.

Algorithm 3: Stepwise linear function for sensitivity validation

Input : x_1
Input : x_2
if $x_1 < 0$ **then**
  $y \leftarrow x_2$;
end
if $x_1 \geq 0$ *and* $x_1 < 2$ **then**
  $y \leftarrow (x_2 + 3)$;
end
if $x_1 \geq 2$ **then**
  $y \leftarrow (x_2 + 6)$;
end
return y ;

5.2.2 Iris dataset

A simple MLP model was trained on the Iris dataset (Fisher, 1936). SHAP was used to generate explanations, following the guidance provided in the SHAP documentation.

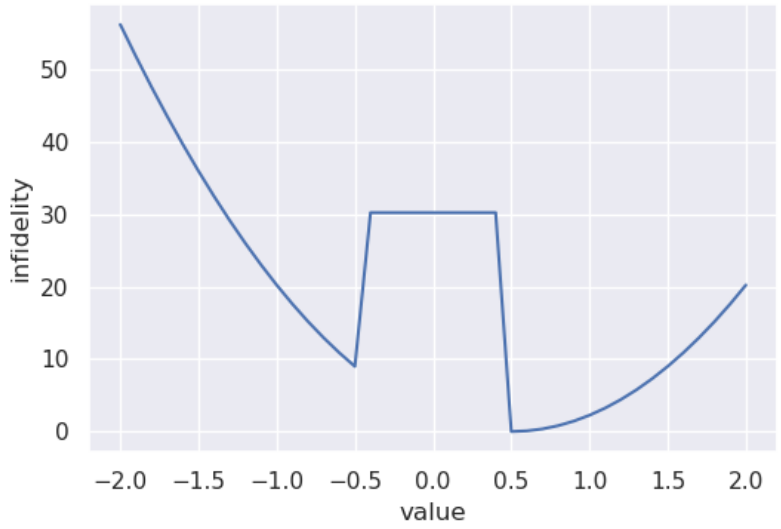

FIGURE 5.2: Set $e_2 = 0.5$, and vary e_1 between `[-2, 2]`. This was expected to show the same result as Figure 5.1, except for between `[0.5, 0.5]`, where the result should be constant. This is because between these values, the algorithm should class e_2 as the most important feature (as has highest absolute value). This result is in line with the expectation.

These explanations were then measured for sensitivity and infidelity in a systematic way as described below.

For sensitivity, the notion of 'prediction uncertainty' was introduced. This is the degree to which the model is unsure what the correct class is for an instance. In theory, a higher uncertainty value should lead to higher explanation sensitivity. Figure 5.5 and Figure 5.6 show this to be the case in practice, representing a positive result.

For infidelity, random noise was added to explanations for five instances, and the changes to infidelity measured. The expected result was that infidelity should increase as noise increases. Figure 5.7 shows this to be true for four out of five instances.

The instance with an apparently constant value is an interesting case. If we look at this instance specifically and alter the scale of the y-axis (Figure 5.8), it can be seen this exhibits proportionally the same behaviour as the other four instances, but on a scale approximately two orders of magnitude smaller.

Why is this instance different? It has already been shown (in Figure 5.1) that infidelity is low when instances are near the baseline value. Figure 5.9 shows the cumulative difference from baseline values for each of these five instances, and instance with index

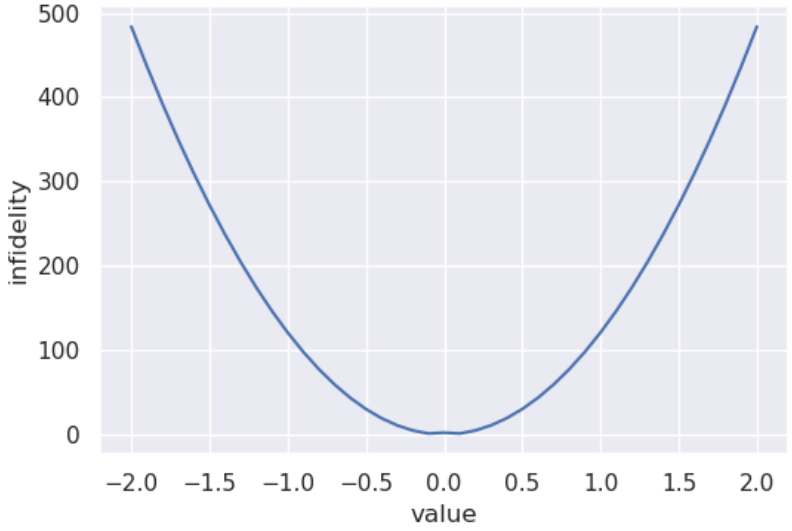

FIGURE 5.3: Set $e_1 = 0$ and vary e_2 between `[-2, 2]`. We expect this to show higher levels of infidelity than for Figure 5.1 away from the baseline, as it is unfaithful to the underlying model to suggest e_2 has a high importance (in fact it has no impact on the model output). Infidelity values are approximately an order of magnitude higher than in Figure 5.1, so this result is in line with expectations.

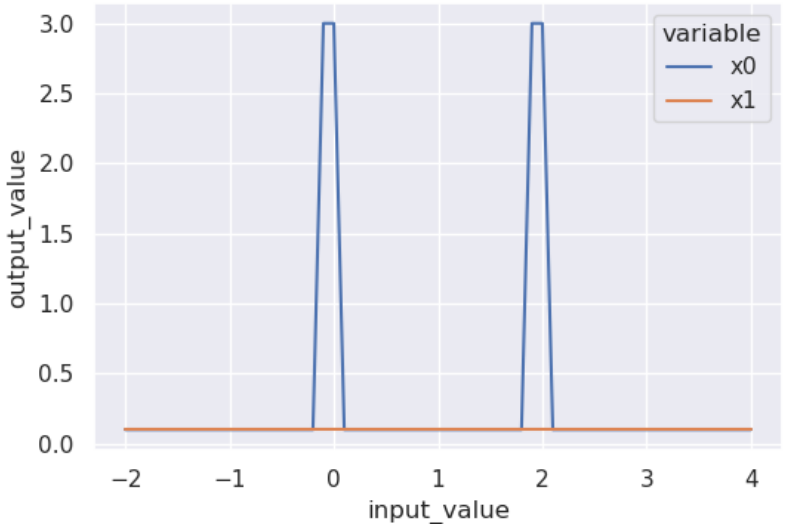

FIGURE 5.4: x_0 and x_1 are independently varied between `[-2, 2]`, with the other value being held constant. As expected, we see spikes in sensitivity for x_0 around the discontinuities at 0 and 2 shown in Algorithm 3

1 having a significantly smaller total than the other instances. Therefore, this seems to be the same behaviour as exhibited by the linear model.

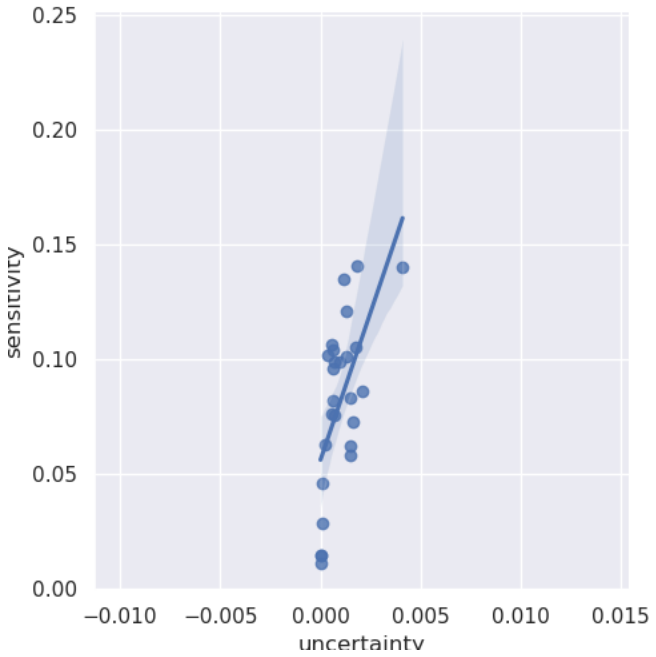

FIGURE 5.5: Sensitivity values for Iris instances plotted against prediction uncertainty, showing a clear positive correlation for instances with low uncertainty.

5.2.3 Metric validation conclusion

These validation exercises have provided useful data on the metrics, and have partially validated Hypothesis 1, with one significant caveat around the infidelity metric. This is the phenomenon where values are lower around the baseline, and are always zero when the instance exactly equals the baseline.

It is unclear how much of an issue this will be in practice - models with larger feature sets will result in the algorithm perturbing more features, all of which are unlikely to be around the baseline values. This is, however, an area for possible future work in evaluating alternative ways of setting baseline values.

5.3 Completeness property of explanations

This section aims to validate Hypothesis 2.

Before considering the theory, we attempt to assess whether the completeness property holds for SHAP experimentally. As documented in Section 2.2.1, 'Completeness' means any change in the model output will be matched with an equal change in the sum of explanation importances generated by the explanation model.

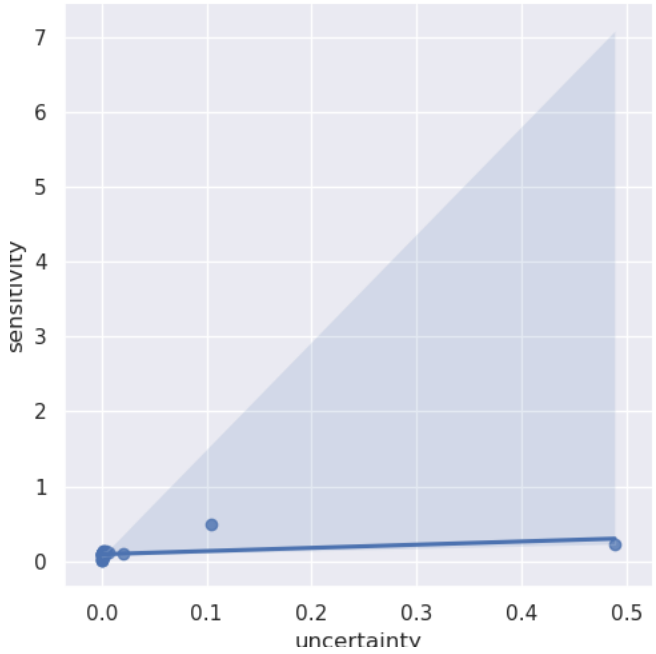

FIGURE 5.6: Positive correlation is less clear for the full dataset, however the number of instances with high uncertainty is limited due to Iris being a relatively simple dataset.

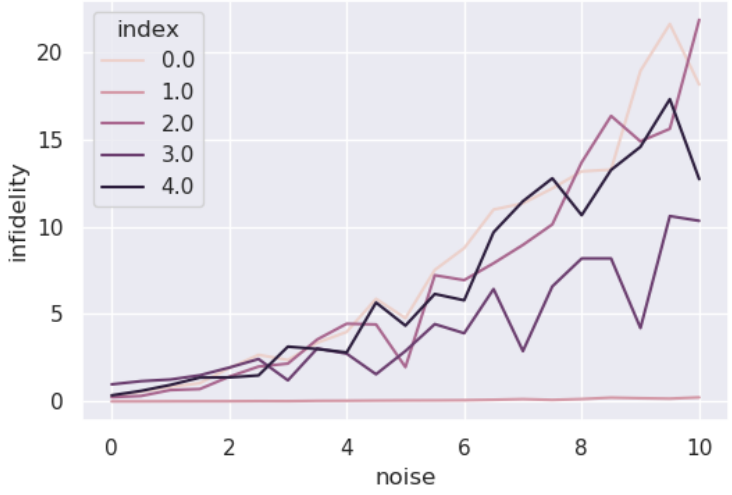

FIGURE 5.7: Iris infidelity measurements for a set of five instances. Four out of the five follow the same, expected, pattern of infidelity increasing as noise is added to the explanation. One appears to stay constant.

A simple experiment was set up to calculate a prediction and explanation for an arbitrary baseline instance, with the Spambase dataset having been used to build an MLP model in TensorFlow (refer to **??** for details of the model). Predictions and explanations were then calculated for 20 other instances. Explanation importances were summed, and plotted against the predictions for each instance. For the completeness property to hold, this chart should show a 1:1 linear relationship, intersecting `(0, 0)`.

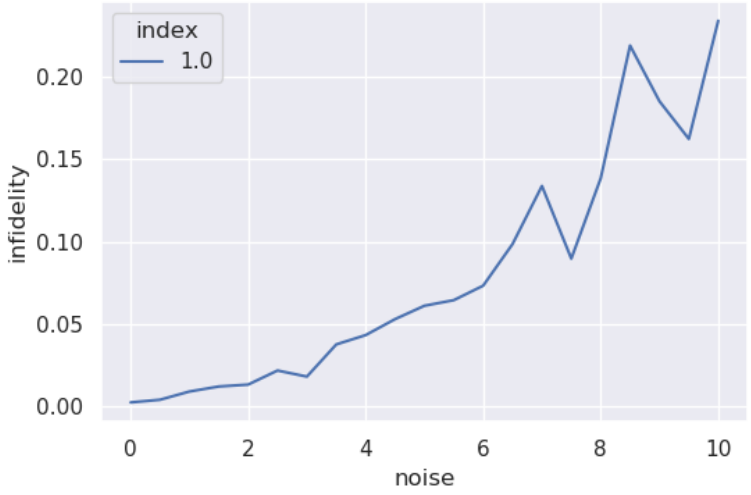

FIGURE 5.8: Iris infidelity measurements for a single instance showing lower infidelity in Figure 5.7

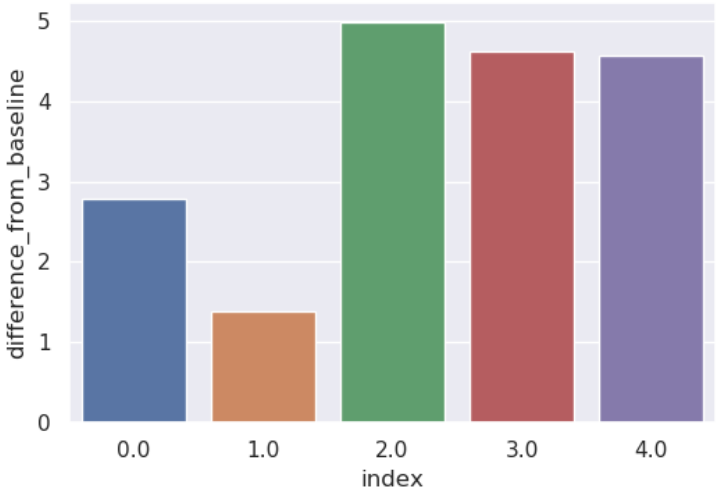

FIGURE 5.9: Total difference from baseline values for the five instances shown in Figure 5.7

Figure 5.10 shows this property does hold for SHAP explanations.

Figure 5.11 was generated by adding large amounts of random noise to each instance (each feature altered by multiplying between a random variable between [-10000, 10000]. As would be expected, this results in a wider range of predicted values than Figure 5.10, however the completeness property still holds.

Figure 5.12 shows the same experiment as Figure 5.10 for LIME. Here we can see clearly the completeness property does not hold, even before noise is added to the instance.

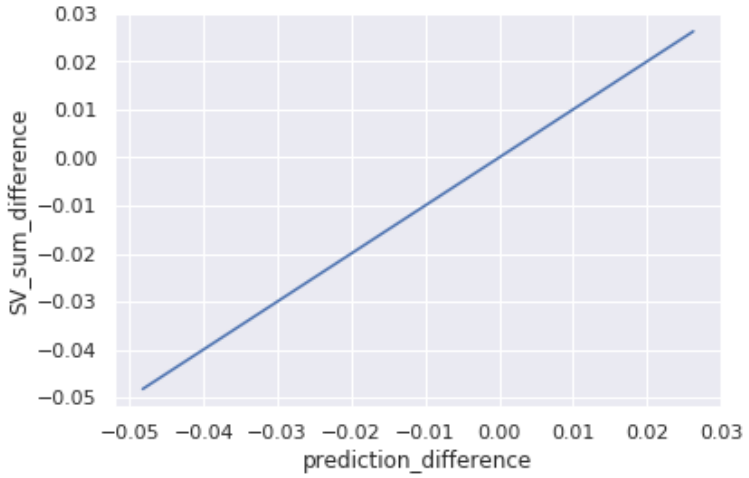

FIGURE 5.10: Validation of completeness property for SHAP with a set of 20 instances

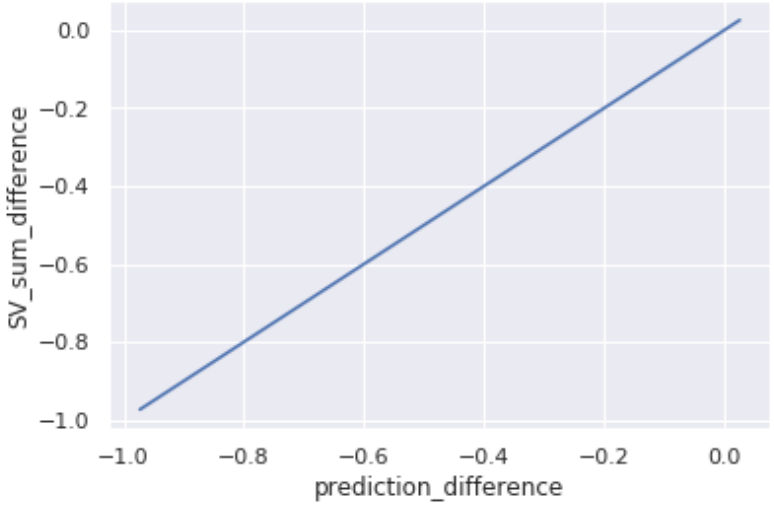

FIGURE 5.11: Validation of completeness property for SHAP with a set of 20 instances, with large amounts of noise added

The difference is due to SHAP having the property of 'local accuracy', as described in the original paper introducing it (Lundberg and Lee, 2017). This paper describes a universal definition for all attribution-based explanation methods, including both LIME and SHAP:

Definition 3. Given a predictor model f, explanation model g, instance input x, simplified input x' with M features that can be mapped back to x via a mapping function h_x for the specific instance, and ϕ representing the feature importance for a specific (simplified) input feature:

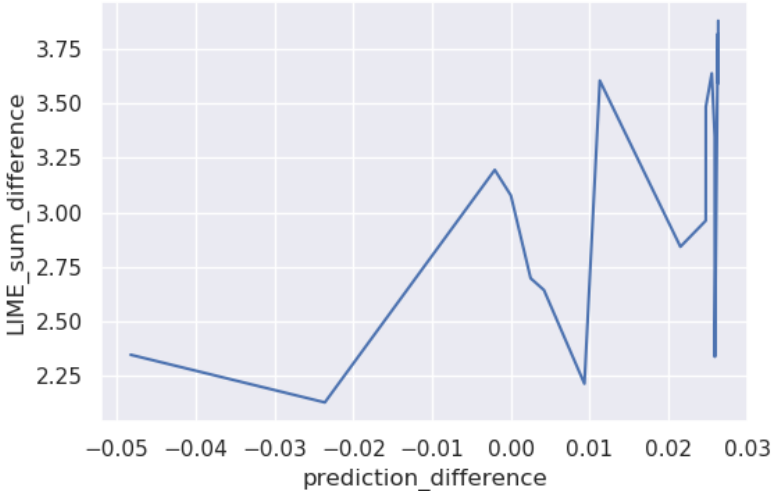

FIGURE 5.12: Failed attempt to validate completeness property for LIME with a set of 20 instances

$$g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$$

Both LIME and SHAP have the concept of simplified input instance (x'). By perturbing the original instance to create new instances, LIME attempts to train a locally interpretable model that uses these simplified instances. This new, separate model is used to provide feature importances, with (as has been seen experimentally) no guarantees of any sort of formal relationship to the original model.

SHAP calculates feature importances in quite a different way. Firstly, it attempts to simplify the feature space into a smaller number of features (k-means is one possible way of clustering features in order to achieve this). The algorithm then systematically works through different 'coalitions' (possible combinations) of these simplified features (x' values), each time measuring the impact on model output by using h_x to map $x \leftarrow x'$. By doing this, it can calculate the average impact of a feature across all different coalitions. It is only at the final stage of the process that these 'average impacts' are used to build a linear model. There is no process of 'training' the model, as is normally understood by the term in ML.

Although LIME and SHAP are superficially similar in that they arrive at a linear model, they take very different approaches, with SHAP having a more rigorous mathematical foundation.

We can say with high confidence Hypothesis 2 is validated, as the empirical results align with the theory. There are clearly some explanation techniques where the completeness property does not hold, and therefore referring to it as an 'axiom' (as in (Yeh et al., 2019)) is misleading. This is a useful finding for building confidence in the output of the infidelity metric (when used with SHAP), and also in allowing the documentation for the `exrt` Python package to be updated to add this caveat for infidelity.

5.4 Adversarial classifier models

The Spambase dataset and model/explanation were used, as documented in **??**.

Figures Figure 5.13 and Figure 5.14 show the results of sensitivity and infidelity for the first ten test set instances for all five model types, as shown in Figure 4.5.

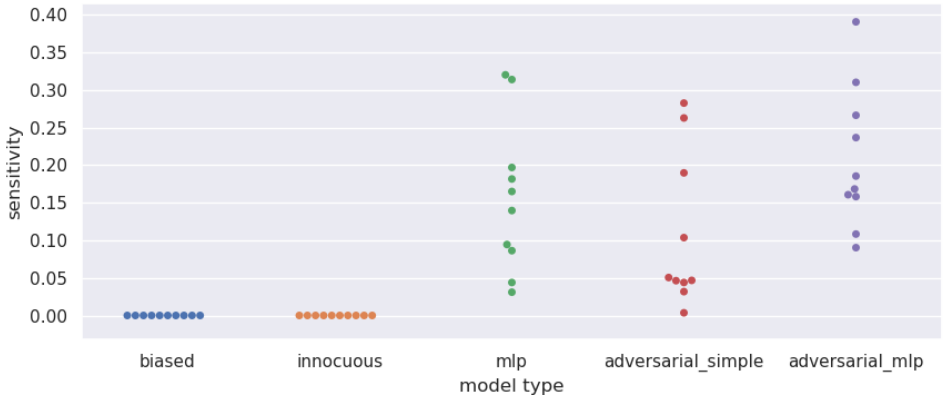

FIGURE 5.13: Measurement of explanation sensitivity for the first ten test set instances of the Spambase dataset. The different models measured are described in Figure 4.5

The most interesting comparison, and the most relevant for validating Hypothesis 3, is between `mlp` and `adversarial_mlp`. As stated in Section 4.3.1, the former is a 'normal' MLP model, and the latter is designed to give a false explanation. As shown earlier, both give the same explanation, but a different prediction. Can our metrics tell them apart?

Based on the infidelity measure alone, they cannot. The majority of instances have a very low value for infidelity, with a handful for each model having a higher value. This is a surprising result - I expected this metric to be able to tell the false explanation is not faithful to the underlying model.

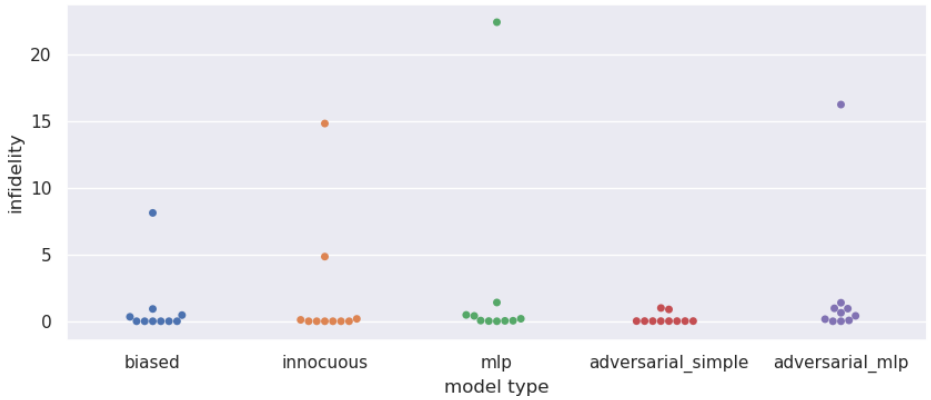

FIGURE 5.14: Measurement of explanation infidelity for the first ten test set instances of the Spambase dataset. The different models measured are described in Figure 4.5

Sensitivity shows more promise here. Figure 5.13 shows a clear distinction between the two relevant model types. The average value for `mlp` is 0.16; for `adversarial_mlp` the average is 0.21. This result is more in line with expectations - we would expect our adversarial model to be more sensitive, as there are in effect two models wrapped into one, either of which can exhibit a lack of robustness.

Overall, the results for Hypothesis 3 are inconclusive. While sensitivity shows signs of being able to distinguish the different models, the relationship is not strong enough to validate this hypothesis. Further work could look to determine why infidelity could not tell the different models apart.

5.5 Overfitted models

The Bank Marketing dataset and model/explanation were used, as documented in **??**. Ten random test set instances were input to the predictor model, and had explanations generated using SHAP. Sensitivity and infidelity were then calculated using the `exrt` Python package published as part of this project. The results are summarised in Figure 5.15. This shows a clear difference in sensitivity between the baseline and overfitted model. This is a strong result, and clearly shows this metric is able to distinguish an explanation generated from an overfitted model.

The results for infidelity are less clear. As shown in Figure 5.16, most instances have a value approaching zero, with two outliers for the overfitted model with higher values.

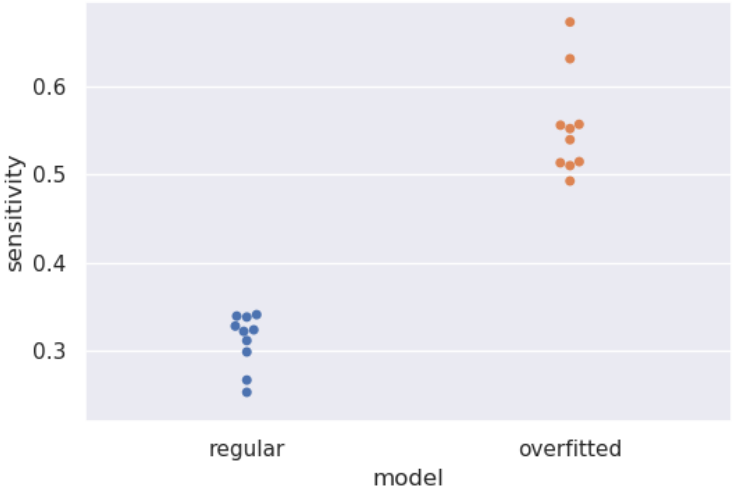

FIGURE 5.15: Results of sensitivity analysis for bank marketing dataset, showing the overfitted and baseline model are clearly separable by this metric.

To ensure this is not a coincidence, the experiment was re-run for infidelity only with a large sample size, results are shown in Figure 5.17. This demonstrates the result was not coincidence, with several instances from the overfitted dataset having high values for infidelity.

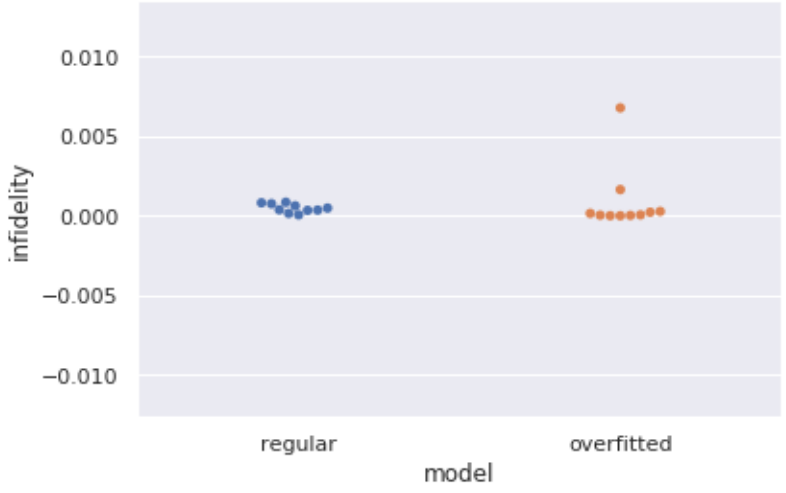

FIGURE 5.16: Results of infidelity analysis for bank marketing dataset, showing most instances have a value of or near zero, with two outliers for the overfitten models.

These results are useful, in that they show the metrics are helpful in telling us when explanations may not be robust due to an unreliable model. Sensitivity appears to work well in telling us properties about the overall model, and infidelity can tell us if an

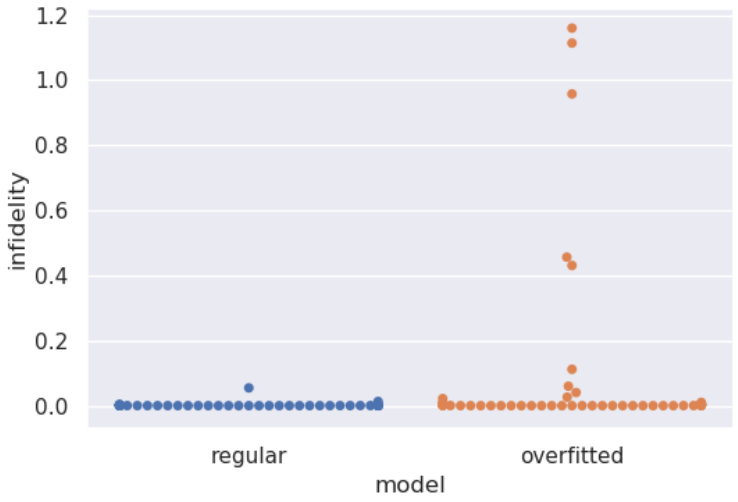

FIGURE 5.17: Results of infidelity analysis for bank marketing dataset with a larger sample size. Demonstrates the 'outliers' noted in Figure 5.16 are in fact part of a trend.

explanation for a specific instance is unreliable. However, they do not in themselves validate Hypothesis 3, which splits into two parts:

- *Sensitivity and fidelity measure explanations of overfitted models as having low robustness* - this has been successfully demonstrated.
- *... when the explanations are in fact consistent with the model* - this has not yet been demonstrated.

It is unclear exactly how to demonstrate an explanation is 'in fact consistent' with the model, aside from using the very techniques we are seeking to assess here. One proxy might be to measure whether there is a correlation between model and explanation robustness. A strong correlation would suggest (but not prove) that model robustness is influencing explanation robustness.

To measure this, a simple function for assessing model robustness was written. This carries out the same perturbations as the sensitivity calculation, but measures the difference in model prediction, rather than the difference between explanations.

Figure 5.18 shows this model robustness plotted against sensitivity. As expected, the overfitted model shows very low model robustness. This suggests (but does not prove) the low model robustness may be driving the high explanation sensitivity.

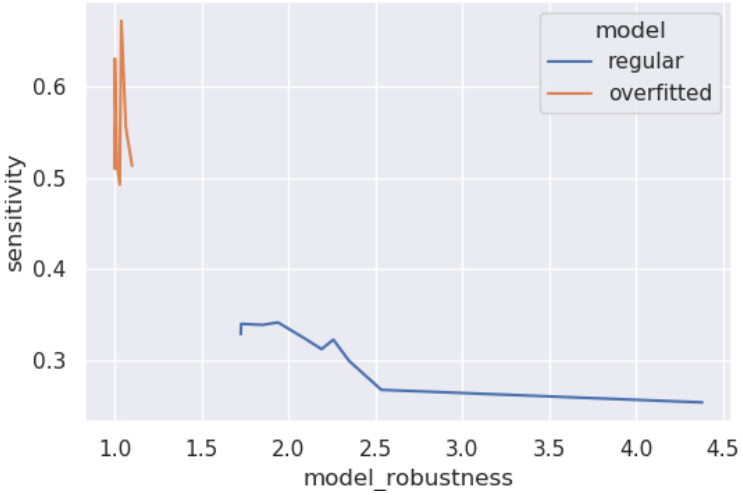

FIGURE 5.18: Results of comparing model robustness with sensitivity for bank dataset.

In summary, the results for Hypothesis 4 are inconclusive, although this section has demonstrated the metrics can tell use useful properties about explanation and instances - namely that they were generated by an overfitted model.

Chapter 6

Conclusion

6.1 Summary

As noted in Section 1.3, this project has met all of its original objectives (although one with slightly reduced scope), and all high priority requirements as described in Section 3.3. It makes a twofold contribution to the field:

- **Software**: a working 'proof of concept' Python package `exrt` that allows explanations of ML models to be measured for robustness in two ways - their sensitivity and infidelity.
- **Experimental**: successful validation of these metrics using linear models and the Iris dataset.

Another useful, and unexpected, contribution in this work is extending the work on adversarial models by (Slack et al., 2019) to cover neural network/MLP models, which provides a more realistic 'false' explanation - refer to Figure 4.7 and Figure 4.8 for a comparison. This makes the technique potentially more difficult to detect in a real-world setting.

Attempts to formally prove my hypotheses were mixed. As noted in Section 6.3, my third and fourth hypotheses could have been more clearly and simply written.

- Hypothesis 1: validation of metrics. **Largely proven**, but with one caveat around he infidelity metric for instances close to baseline values. This issue could be why infidelity is generally the weaker of the two metrics, giving less conclusive results.
- Hypothesis 2: proving completeness holds for SHAP, but not universally for all XAI techniques. **Successfully proven**. This is a useful finding that proves this property should not be described as an 'axiom', and provides important limitations for use of the metrics.
- Hypothesis 3: metrics can detect adversarial models. **Inconclusive**. Sensitivity metric can detect a slight difference. No discernible difference for the infidelity metric, which is a surprising result.
- Hypothesis 4: metrics can detect overfitted models. Both metrics show a clear ability to detect overfitted models. However, I was unable to prove the explanations are consistent with the model, so this is **inconclusive**.

With hindsight, it is perhaps unsurprising the last two hypotheses were a null result. The results of these metrics depend on a number of complex factors in addition to the explanation robustness they seek to measure - the underlying model robustness, the specific instance, the baseline values provided and the data types (numerical, nominal, ordinal).

These metrics are useful in their current state as a 'sense check' for explanations - for example, comparing different XAI techniques to see if any have particularly high sensitivity, suggesting they are not to be relied upon. Or, testing each instance fed into an explainer function to see if any have particularly high infidelity values, again suggesting the explanation of that instance should be used with caution.

However, more work is needed before these metrics could be used more formally, for example in assessing robustness of explanations as required by legislation.

6.2 Future work

I firmly believe the level of interest in XAI will continue to increase in future, as the general public become more aware of the pervasiveness of ML techniques, and their lack

of transparency. I hope we will either see a shift away from 'black box' ML techniques, and towards inherently interpretable models (such models continue to improve their performance, and are often at the same level as DNNs in real-world situations (Rudin, 2018)). Or, that XAI techniques improve to the point they can be relied upon to give robust and accurate results, and are easily able to detect bias and adversarial techniques.

With further refinements, the metrics I have published could make a small but useful contribution to the second of these approaches. I plan to carry on working on the software as an open-source package, and will prioritise the following areas:

- Resolving the issue with infidelity of instances near the baseline values giving a low result. A non-deterministic, sampling-based approach is perhaps needed.
- Improving performance of the sensitivity metric - or at least adding further hyper-parameters so accuracy and performance can be more explicitly traded off.
- Further assessments using 'real-life' models. Perhaps with human involvement (i.e. compare human and AI driven assessment of explanation robustness).

6.3 Reflections and lessons learnt

I am pleased I chose to focus on this area. I knew nothing about XAI when I started, let alone how to assess explanations for robustness. Building knowledge in these areas has been a significant challenge, especially given this is a young field, with inconsistent terminology in the literature. I have learnt a huge amount from this project - technical knowledge of ML and XAI, academic communication in oral and written form, and project/time management when working alone.

With hindsight, I would have changed my approach to the project in certain areas:

- **Hypotheses** - make these simpler, and follow the *If [I do this] then [this will happen]* format.
- **Speed of experiments** - more up front investment would have made my work faster, in particular investing in a cloud-based parallel computing environment.

- **Preliminary experimentation** - I should have invested more time in preliminary investigation of the metrics, and less time on the literature review. This would have resulted in more precise hypotheses, as I built an understanding of the limitations of these metrics.

Appendix A

Professional, Legal, Ethical, and Social Issues

This is the PLESI section of my Research Report.

A.1 Professional Issues

I will publish code in a GitHub repository alongside my dissertation. Software developed will be of a high standard - well structured and documented. Taking inspiration from https://paperswithcode.com/, I will publish a summary Jupyter Python notebook which allows my results to be easily reproduced in a step-by-step manner, while offering a brief summary of the theory.

This software will be developed in the open, in a public repository. This will make it simpler to obtain regular feedback on the code. I have confirmed with my project supervisor this approach is acceptable.

The software documentation will make it clear what the software does and does not do. It will clearly describe any uncertainties and limitations.

Work by third parties (e.g. external libraries) will be credited in the code README, as well as in my final report. I will credit others' work even when not legally required by licences.

A.2 Legal Issues

Any software developed will be licensed under the **GNU General Public License v3.0**. This licence has been chosen for three reasons:

1. Limitation of liability.
2. Allows onward distribution and re-use.
3. Does not permit closed-source usage.

Only datasets commonly used in machine learning research will be used, after ensuring I have the rights to use them, and that they do not contain any personal data. In practice this the most likely be from the UCI datasets: `https://archive.ics.uci.edu/ml/datasets.php`.

A.3 Ethical Issues

Some work (Slack et al., 2019) has focussed specifically on maliciously designed, biased classifiers, that focus on protected characteristics such as race and sex to make decisions. The paper cited concerns how this behaviour could be obfuscated by manipulating explanations.

In this work, I do not plan to focus explicitly on bias. it is worth nothing however that the techniques I am investigating could potentially be used as a defence against such manipulation techniques; whether for obscuring biased classifiers, or any other unwanted behavior.

If successful, I hope my work may make a small positive contribution to the discussion of how to assure robustness of ML explanations, including those that concern issues of fairness and bias.

A.4 Social Issues

Refer to Appendix A.5.

A.5 Review against ORBIT AREA 4P Framework

In addition to considering the above areas, a review against the ORBIT AREA 4P Framework (Orbit RRI, 2018) was carried out. This is an ethics framework specifically tailored to IT projects. Two questions it poses are relevant for this project:

- **Have we included the right stakeholders?** One advantage of this project is it being part of the Lab for AI Verification at Heriot-Watt University. This has allowed me to obtain feedback from a wider group of academics than just my supervisor; both from weekly meetings, and a presentation to the whole lab group. This experience has been extremely valuable.
- **Will the products be socially desirable?** The 'product' in this case - if a positive result is obtained - is a novel technique for assuring the robustness of ML explanations. Such a product is clearly socially desirable - particularly given the possibility of ML models being biased or exhibiting other undesirable behaviour.

Appendix B

Project Management

This is the project management section of my Research Report.

B.1 Risk Management

A list of project risks and mitigations is given in Table B.1. Note that mitigations do not reduce the level of risk to zero.

As noted in the table, the main approach to de-risking this project is to prioritise the most challenging/risky areas first (including my preliminary investigations. This will mean any difficulties can be identified and discussed with my project supervisor as soon as possible.

B.2 Project Plan

My project does not involve any human evaluation. Therefore, I am able to build some flexibility into the plan, as it does not need to be centred around this evaluation.

I will divide the project into two distinct phases:

- **Phase 1 (to end May)**: building an MVP following a waterfall approach. See Figure B.1 for a summary of timescales. This will an deliver an assessment of Hypotheses 1 and 2, some simple experiments of heuristic techniques for explanation

robustness (Hypothesis 3), and a basic version of the proposed software package (Objective 7). It will also prove out the scalability and interfaces of the Marabou formal verification tool.

- **Phase 2 (June/July)**: using agile techniques to improve the MVP project. Agreeing the priorities for the next week's 'sprint' with my project supervisor. This phase does not have a detailed plan, although I will aim to deliver all requirements, including those with 'Low' or 'Medium' priority.

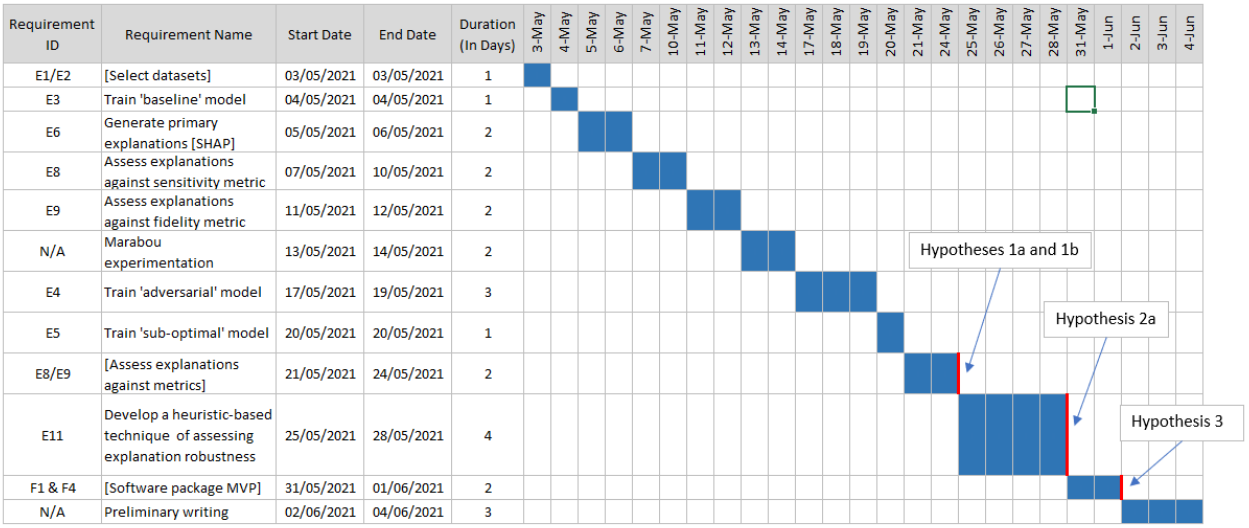

FIGURE B.1: Summary of project plan for Phase 1. Red lines show points where I aim to have investigated different hypotheses.

Table B.1: Risk management approach

Risk	**Prob.**	**Impact**	**Mitigation**	**Prob. after mitigation**	**Impact after mitigation**
Unable to replicate results of other studies (particularly those involving explanation metrics)	Medium	Medium	Prioritise this task in the project plan. Establish contact with authors of studies if required.	↓ Low	Medium
Unable to find suitable datasets to base the project around	Low	High	Prioritise this task in the project plan.	↓ Negligible	High
Covid-19 causes further school/nursery closures in Scotland - therefore unable to work full-time	Low	High	Adjust project scope and use mitigating circumstances procedure if required.	Low	↓ Medium
Loss of access to GitHub/Overleaf	Low	High	Set up link between Overleaf (LaTeX) and GitHub. Make sure GitHub repository also stored locally.	Low	↓ Low
Formal verification using Marabou does not scale to DNNs	Medium	Medium	Prioritise this task in the project plan.	↓ Low	Medium
Failure to meet deadline	Low	High	Adjust project plan to build a 'minimal viable product' execution of the project first, before iterating on this as time allows. Build contingency into the plan.	↓ Negligible	High
Unable to disprove null hypotheses	Medium	Medium	Ensure attending all supervision meetings, and make early decisions on whether to carry on experimenting.	↓ Low	Medium
Inadvertent plagiarism due to lack of literature review coverage	Medium	High	Treat literature review as a continuous exercise. Set up Google Scholar alerts to notify me if any new relevant work published.	↓ Low	High
Experiments too slow due to computational constraints	Medium	High	Reconsider approach - make sure parallelising and using GPU. Use cloud computing options if required.	↓ Low	High

Appendix C

Changes to Methodology

This appendix summarises changes made to the methodology of this project since the research report was written. All changes were made during the implementation phase, in response to time constraints, or in order to investigate interesting findings further. Changes were not made in order to achieve more desirable results.

Functional software requirements remain the same. Some changes have been made to experimental requirements. In summary, I found there was a large amount of interesting and novel analysis to be carried out on the existing metrics, and decided to focus on this rather than attempting to find new metrics.

C.1 Added since Research Report

Two new requirements, both of which can be seen in Table 3.2:

- **E15**: Assess completeness property of explanations
- **E16**: Implement sensitivity and infidelity metrics (previously I assumed I would be able to reuse the existing implementation by (Yeh et al., 2019). This was not the case; I started from scratch in order to build an improved version of the metrics).

For my hypotheses, 1 and 2 are both new, and assess the results of implementing the above two requirements.

C.2 Removed since Research Report

Hypothesis 3. *Heuristic techniques, such as combining and comparing different explanations, can be used to design measures of explanation robustness which improve on existing measures in an environment where model robustness is uncertain.*

Hypothesis 4. *Formal verification techniques can be used to design measures of explanation robustness which improve on existing measures in an environment where model robustness is uncertain.*

The following requirements were also removed:

- **E1** Select dataset for regression problem.
- **E7** Generate secondary explanations.
- **E11** Develop a heuristic-based technique of assessing explanation robustness.
- **E12** Develop a formal verification-based technique of assessing explanation robustness.

All hypotheses and requirements were removed due to time constraints, and the extra work added as described above.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X., and Brain, G. (2016). Tensorflow: A system for large-scale machine learning. pages 265–283. {USENIX} Association.

Alaa, A. M. and Schaar, M. V. D. (2019). Demystifying black-box models with symbolic metamodels.

Alvarez-Melis, D. and Jaakkola, T. S. (2018). On the robustness of interpretability methods. *arXiv*.

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016). Machine bias — ProPublica. https://www.propublica.org/article/bias-in-criminal-risk-scores-is-mathematically-inevitable-researchers-say - Accessed on 04.04.2021.

Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M., and Eckersley, P. (2020). Explainable machine learning in deployment. pages 648–657. Association for Computing Machinery, Inc.

Calcagno, C., Distefano, D., Dubreil, J., Gabi, D., Hooimeijer, P., Luca, M., O'Hearn, P., Papakonstantinou, I., Purbrick, J., and Rodriguez, D. (2015). Moving fast with software verification. volume 9058, pages 3–11. Springer Verlag.

Caliskan, A., Bryson, J. J., and Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science*, 356:183–186.

Carbonell, J. R. (1970). Ai in cai: An artificial-intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11:190–202.

Casadio, M., Daggitt, M. L., Komendantskaya, E., Kokke, W., Kienitz, D., and Stewart, R. (2021). Property-driven training: All you (n)ever wanted to know about. *arXiv*.

Chan, A., Tay, Y., Ong, Y. S., and Fu, J. (2019). Jacobian adversarially regularized networks for robustness. *arXiv*.

Chollet, F. et al. (2015). Keras. https://keras.io.

Dombrowski, A.-K., Alber, M., Anders, C. J., Ackermann, M., Müller, K.-R., and Kessel, P. (2019). Explanations can be manipulated and geometry is to blame. *arXiv*.

D'Silva, V., Kroening, D., and Weissenbacher, G. (2008). A survey of automated techniques for formal software verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27:1165–1178.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Dzindolet, M. T., Peterson, S. A., Pomranky, R. A., Pierce, L. G., and Beck, H. P. (2003). The role of trust in automation reliance. *International Journal of Human Computer Studies*, 58:697–718.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188.

Flores, A. W., Bechtel, K., Lowenkamp, C. T., Bonta, J., Cullen, F., Latessa, E., Monahan, J., Serin, R., Skeem, J., and Buck, S. (2016). False positives, false negatives, and false analyses: A rejoinder to "machine bias: There's software used across the country to predict future criminals. and it's biased against blacks.".

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. International Conference on Learning Representations, ICLR.

Goodman, B. and Flaxman, S. (2017). European union regulations on algorithmic decision making and a "right to explanation". *AI Magazine*, 38:50–57.

Heo, J., Joo, S., and Moon, T. (2019). Fooling neural network interpretations via adversarial model manipulation. *arXiv*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv*.

Ignatiev, A. (2020). Towards trustable explainable ai. volume 2021-January, pages 5154–5158. International Joint Conferences on Artificial Intelligence Organization.

Ignatiev, A., Narodytska, N., and Marques-Silva, J. (2019). On relating explanations and adversarial examples. volume 32.

Jo, J. and Bengio, Y. (2017). Measuring the tendency of cnns to learn surface statistical regularities. *arXiv*.

Julian, K. D., Sharma, S., Jeannin, J.-B., and Kochenderfer, M. J. (2019). Verifying aircraft collision avoidance neural networks through linear approximations of safe regions. *arXiv*.

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., Dill, D. L., Kochenderfer, M. J., and Barrett, C. (2019). The marabou framework for verification and analysis of deep neural networks. volume 11561 LNCS, pages 443–452. Springer Verlag.

Kokke, W., Komendantskaya, E., Kienitz, D., Atkey, R., and Aspinall, D. (2020). Neural networks, secure by construction an exploration of refinement types.

Komendantskaya, E., Kokke, W., and Kienitz, D. (2020). Continuous verification of machine learning: a declarative programming approach.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. volume 25.

Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial machine learning at scale. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Lechner, M., Hasani, R., Grosu, R., Rus, D., and Henzinger, T. A. (2021). Adversarial training is not ready for robot learning.

Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551.

Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., and Kochenderfer, M. J. (2019). Algorithms for verifying deep neural networks. *arXiv*.

Lundberg, S. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 2017-December:4766–4775.

Moro, S., Cortez, P., and Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31.

Orbit RRI (2018). AREA 4P Framework. https://www.orbit-rri.org/about/area-4p-framework/ - Accessed on 04.04.2021.

PwC (2018). Explainable AI. https://www.pwc.co.uk/audit-assurance/assets/pdf/explainable-artificial-intelligence-xai.pdf - Accessed on 04.04.2021.

PYpI (2021). The Python Package Index. https://pypi.org/-Accessedon15.08.2021.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?" explaining the predictions of any classifier. volume 13-17-August-2016, pages 1135–1144. Association for Computing Machinery.

Ribera, M. and Àgata Lapedriza (2019). Can we do better explanations? a proposal of user-centered explainable ai.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Rozenblit, L. and Keil, F. (2002). The misunderstood limits of folk science: an illusion of explanatory depth. *Cognitive Science*, 26:521–562.

Rudin, C. (2018). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215.

Samek, W. (2019). *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700. Springer International Publishing.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G. V. D., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu,

K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489.

Sitawarin, C., Bhagoji, A. N., Mosenia, A., Chiang, M., and Mittal, P. (2018). Darts: Deceiving autonomous cars with toxic signs. *arXiv*.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. (2019). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. *AIES 2020 - Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. International Conference on Learning Representations, ICLR.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2018). Robustness may be at odds with accuracy. *arXiv*.

Vreš, D. and Šikonja, M. R. (2020). Better sampling in explanation methods can prevent dieselgate-like deception.

Yang, F., Du, M., and Hu, X. (2019). Evaluating explanation without ground truth in interpretable machine learning. *arXiv*.

Yeh, C.-K., Hsieh, C.-Y., Suggala, A. S., Inouye, D. I., and Ravikumar, P. (2019). On the (in)fidelity and sensitivity of explanations. volume 32.

Zhang, Q., Wu, Y. N., and Zhu, S.-C. (2017). Interpretable convolutional neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8827–8836.

Zhao, Z., Dua, D., and Singh, S. (2017). Generating natural adversarial examples. *arXiv*.